

Fire Programmer's Reference Manual

For Fire 2.1

Release Date: July 30, 2007





Products Rights Notice:

Copyright © 1991-2007 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, California 95054, U.S.A. All Rights Reserved

You understand that these materials were not prepared for public release and you assume all risks in using these materials. These risks include, but are not limited to errors, inaccuracies, incompleteness and the possibility that these materials infringe or misappropriate the intellectual property right of others. You agree to assume all such risks.

THESE MATERIALS ARE PROVIDED BY THE COPYRIGHT HOLDERS AND OTHER CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS (INCLUDING ANY OF OWNER'S PARTNERS, VENDORS AND LICENSORS) BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THESE MATERIALS, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. Sun, Sun Microsystems, the Sun logo, Solaris, OpenSPARC T1, OpenSPARC T2 and UltraSPARC are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon architecture developed by Sun Microsystems, Inc. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd. The Adobe logo is a registered trademark of Adobe Systems, Incorporated. Part of the products covered by these materials may be derived from the Berkeley BSD systems licensed by the University of California. Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product described in these materials. This distribution may include materials developed by third parties who have intellectual property rights therein. Products covered by and information contained in these materials may be controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or re-export to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists may be prohibited

1.1	Introduction	20
1.1.1	Chapter Organization	20
1.1.2	Fire Overview	20
1.1.3	Fire Block Diagram	22
1.2	Operational Overview	23
1.2.1	Fire Overview	23
1.2.2	JBus to Fire (Downbound) Transactions	23
1.2.2.1	JBUS Physical Address Partitioning	23
1.2.2.2	Offset Base and Offset Mask Register Behavior	27
1.2.2.3	8 MByte Noncacheable Configuration Region	28
1.2.2.4	Ebus Sub-region	29
1.2.2.5	I2C Region	30
1.2.2.6	PCI Express Configuration & I/O Sub-region	30
1.2.2.7	PCI Express 32-bit Addressing Memory Sub-region	31
1.2.2.8	PCI Express 64-bit Addressing Memory Sub-region	32
1.2.2.9	JBus Interrupt Space	33
1.2.3	Fire to JBus (Upbound) Transactions	33
1.2.3.1	PCI Express => Fire	33
1.2.4	MMU and Bypass Operation	36
1.2.4.1	Translation Storage Buffer Overview	36
1.2.4.2	Translation Table Entry (TTE)	37
1.2.5	Bypass and MMU Initialization	38
1.2.5.1	TSB Cache Operation, Initialization, and Debug	39
1.2.6	Interrupt Model	41
1.2.6.1	Internal Interrupts	42
1.2.6.2	External Out of Band Interrupts	42
1.2.6.3	PCI Express INTx Emulation	42
1.2.6.4	Event Queue Interrupts	43
1.2.6.5	JBUS Interrupt Mondos	45
1.2.6.6	Interrupt Relocation	46
1.2.6.7	Data flushing	47



1.2.6.8	Event Queue Records	47
1.2.7	EStar Mode	50
1.2.8	PCI Power Management	51
1.2.9	Endianness	52
1.2.10	Error Register Overview	53
1.2.11	Performance Register Overview	55
1.2.12	Link Layer Thresholds	56
1.2.12.1	Ack Latency Timer Threshold	56
1.2.12.2	Frequent Nak Latency Timer Threshold	56
1.2.12.3	Replay Timer Threshold	57
1.2.13	Link Ingress Initial Posted Data Credit	58
1.2.14	Register Reset Behavior and SW Access to Fire	58
1.3	CSR Fields and Bits	59
1.3.1	General Information	59
1.3.1.1	Access Size	59
1.3.1.2	Unimplemented Addresses	59
1.3.1.3	Physical Addresses	59
1.3.2	Register Map	60
1.3.3	JBus Registers	79
1.3.3.1	JBus Device ID Register (0x00000000 / 0xFC00000000390000)	79
1.3.3.2	EBus Offset Base Register (0x00400020 / 0x80000000FF000000)	81
1.3.3.3	EBus Offset Mask Register (0x00400028 / 0x7FFF80000000)	81
1.3.3.4	PCIE-A Mem32 Offset Base Register (0x00400040 / 0x0)	82
1.3.3.5	PCIE-A Mem32 Offset Mask Register (0x00400048 / 0x7F0000000000)	83
1.3.3.6	PCIE-A Cfg/IO Offset Base Register (0x00400050 / 0x0)	83
1.3.3.7	PCIE-A Cfg/IO Offset Mask Register (0x00400058 / 0x7F0000000000)	84
1.3.3.8	PCIE-B Mem32 Offset Base Register (0x00400060 / 0x0)	85
1.3.3.9	PCIE-B Mem32 Offset Mask Register (0x00400068 / 0x7F0000000000)	85
1.3.3.10	PCIE-B Cfg/IO Offset Base Register (0x00400070 / 0x0)	86

1.3.3.11	PCIE-B Cfg/IO Offset Mask Register (0x00400078 / 0x7F00000000)	87
1.3.3.12	PCIE-A Mem64 Offset Base Register (0x00400080 / 0x0) . . .	87
1.3.3.13	PCIE-A Mem64 Offset Mask Register (0x00400088 / 0x7F00000000)	88
1.3.3.14	PCIE-B Mem64 Offset Base Register (0x00400090 / 0x0)	89
1.3.3.15	PCIE-B Mem64 Offset Mask Register (0x00400098 / 0x7F00000000)	90
1.3.3.16	Fire Control/Status Register (0x00410000 / 0x7F0038B6000)	90
1.3.3.17	JBus PLL Control and DTL Control Register (0x00410050 / 0x6)	97
1.3.3.18	JBus Energy Star Control Register (0x00410058 / 0x1)	101
1.3.3.19	JBus Change Initiation Control Register (0x00410060 / 0x0)	101
1.3.3.20	Reset Generation Register (0x00417010 / 0x0)	102
1.3.3.21	Reset Source Register (0x00417018 / 0x0)	103
1.3.3.22	GPIO Port 0 Pin 0 Data Register (0x00460000 / 0x0)	105
1.3.3.23	GPIO Port 0 Pin 1 Data Register (0x00460008 / 0x0)	105
1.3.3.24	GPIO Port 0 Pin 2 Data Register (0x00460010 / 0x0)	105
1.3.3.25	GPIO Port 0 Pin 3 Data Register (0x00460018 / 0x0)	106
1.3.3.26	GPIO Port 0 Data Register (0x00460020 / 0x0)	106
1.3.3.27	GPIO Port 0 Control Register (0x00460028 / 0x0)	107
1.3.3.28	GPIO Port 1 Pin 0 Data Register (0x00462000 / 0x0)	107
1.3.3.29	GPIO Port 1 Pin 1 Data Register (0x00462008 / 0x0)	108
1.3.3.30	GPIO Port 1 Pin 2 Data Register (0x00462010 / 0x0)	108
1.3.3.31	GPIO Port 1 Pin 3 Data Register (0x00462018 / 0x0)	108
1.3.3.32	GPIO Port 1 Data Register (0x00462020 / 0x0)	109
1.3.3.33	GPIO Port 1 Control Register (0x00462028 / 0x0)	109
1.3.3.34	EBus EPROM Timing Control Register (0x00464000 / 0xFAD5ABF5F7)	110
1.3.3.35	EBus Chip Select 1 Timing Control Register (0x00464008 / 0xFAD5ABF5F7)	116
1.3.3.36	EBus Chip Select 2 Timing Control Register (0x00464010 / 0xFAD5ABF5F7)	121



1.3.3.37	EBus Chip Select 3 Timing Control Register (0x00464018 / 0xFAD5ABF5F7)	126
1.3.3.38	I2C 0 Input Monitor Register (0x00466000 / 0x0).....	131
1.3.3.39	I2C 0 Data Drive Register (0x00466008 / 0x1).....	131
1.3.3.40	I2C 0 Clock Drive Register (0x00466010 / 0x1).....	132
1.3.3.41	I2C 1 Input Monitor Register (0x00468000 / 0x0).....	132
1.3.3.42	I2C 1 Data Drive Register (0x00468008 / 0x1).....	133
1.3.3.43	I2C 1 Clock Drive Register (0x00468010 / 0x1).....	133
1.3.3.44	PCIE-A Leaf CSR Ring Slow Only Access (0x00470000 / 0x0)	134
1.3.3.45	PCIE-B Leaf CSR Ring Slow Only Access (0x00470008 / 0x0)	134
1.3.3.46	JBus Parity Control Register (0x00470010 / 0x0)	135
1.3.3.47	JBus Scratch Register 1 (0x00470018 / 0x0)	136
1.3.3.48	JBus Scratch Register 2 (0x00470020 / 0x0)	136
1.3.3.49	JBC Error Logic Analyzer Trigger Enable for J_ERR (0x00470028 / 0x3FFFFFFF1FFFFFFF).....	137
1.3.3.50	JBus Scratch Persistent Register (0x00470030 / 0x0)	142
1.3.3.51	JBC Error Log Enable Register (0x00471000 / 0x1FFFFFFF) .	142
1.3.3.52	JBC Interrupt Enable Register (0x00471008 / 0x0)	146
1.3.3.53	JBC Interrupt Status Register (0x00471010 / 0x0).....	150
1.3.3.54	JBC Error Status Clear Register (0x00471018 / 0x0).....	159
1.3.3.55	JBC Error Status Set Register (0x00471020 / 0x0).....	169
1.3.3.56	JBC Fatal Reset Enable Register (0x00471028 / 0x0).....	178
1.3.3.57	JBCINT In Transaction Error Log Register (0x00471030 / 0x0)	179
1.3.3.58	JBCINT In Transaction Error Log Register 2 (0x00471038 / 0x0)	179
1.3.3.59	JBCINT Out Transaction Error Log Register (0x00471040 / 0x0)	180
1.3.3.60	JBCINT Out Transaction Error Log Register 2 (0x00471048 / 0x0)	181
1.3.3.61	Fatal Error Log Register 1 (0x00471050 / 0x0)	182
1.3.3.62	Fatal Error Log Register 2 (0x00471058 / 0x0)	182
1.3.3.63	Merge Transaction Error Log Register (0x00471060 / 0x0) ..	183
1.3.3.64	DMCINT ODCD Error Log Register (0x00471068 / 0x0)....	184

1.3.3.65	DMCINT IDC Error Log Register (0x00471070 / 0x0)	184
1.3.3.66	CSR Error Log Register (0x00471078 / 0x0).	186
1.3.3.67	JBC Core and Block Interrupt Enable Register (0x00471800 / 0x0) 186	
1.3.3.68	JBC Core and Block Error Status Register (0x00471808 / 0x0)	187
1.3.3.69	JBC Performance Counter Select Register (0x00472000 / 0x0)	188
1.3.3.70	JBC Performance Counter Zero Register (0x00472008 / 0x0)	190
1.3.3.71	JBC Performance Counter One Register (0x00472010 / 0x0).	190
1.3.3.72	Fire and JBC Debug Select Register A (0x00473000 / 0x0) . .	190
1.3.3.73	Fire and JBC Debug Select Register B (0x00473008 / 0x0). . .	191
1.3.3.74	I2C Bus-x Slave Address Register (0x00520000, 0x00530000 / 0x0) 192	
1.3.3.75	I2C Bus-x Extended Slave Address Register (0x00520008, 0x00530008 / 0x0)	192
1.3.3.76	I2C Bus-x Data Byte Register (0x00520010, 0x00530010 / 0x0)	193
1.3.3.77	I2C Bus-x Control Register (0x00520018, 0x00530018 / 0x0).	193
1.3.3.78	I2C Bus-x Status Register (0x00520020, 0x00530020 / 0xF8) .	195
1.3.3.79	I2C Bus-x Clock Control Register (0x00520028, 0x00530028 / 0x0) 196	
1.3.3.80	I2C Bus-x Software Reset Register (0x00520030, 0x00530030 / 0x0) 197	
1.3.4	PCI-E Registers	198
1.3.4.1	Interrupt Mapping Registers (0x00601000-0x006011F8, 0x00701000-0x007011F8 / 0x0)	198
1.3.4.2	Interrupt Clear Registers (0x00601400-0x006015F8, 0x00701400- 0x007015F8 / 0x0).	199
1.3.4.3	Interrupt Retry Timer Register (0x00601A00, 0x00701A00 / 0x0) 200	
1.3.4.4	Interrupt State Status Register 1 (0x00601A10, 0x00701A10 / 0x0) 201	
1.3.4.5	Interrupt State Status Register 2 (0x00601A18, 0x00701A18 / 0x0) 201	
1.3.4.6	INTX Status Register (0x0060B000, 0x0070B000 / 0x0)	201
1.3.4.7	INT A Clear Register (0x0060B008, 0x0070B008 / 0x0)	202
1.3.4.8	INT B Clear Register (0x0060B010, 0x0070B010 / 0x0).	203



1.3.4.9	INT C Clear Register (0x0060B018, 0x0070B018 / 0x0)	203
1.3.4.10	INT D Clear Register (0x0060B020, 0x0070B020 / 0x0)	204
1.3.4.11	Event Queue Base Address Register (0x00610000, 0x00710000 / 0x0)	204
1.3.4.12	Event Queue Control Set Register (0x00611000-0x00611118, 0x00711000-0x00711118 / 0x0)	205
1.3.4.13	Event Queue Control Clear Register (0x00611200-0x00611318, 0x00711200-0x00711318 / 0x0)	206
1.3.4.14	Event Queue State Register (0x00611400-0x00611518, 0x00711400-0x00711518 / 0x1)	207
1.3.4.15	Event Queue Tail Register (0x00611600-0x00611718, 0x00711600-0x00711718 / 0x0)	208
1.3.4.16	Event Queue Head Register (0x00611800-0x00611918, 0x00711800-0x00711918 / 0x0)	208
1.3.4.17	MSI Mapping Register (0x00620000-0x006207F8, 0x00720000-0x007207F8 / 0x0)	208
1.3.4.18	MSI Clear Registers (0x00628000-0x006287F8, 0x00728000-0x007287F8 / 0x0)	209
1.3.4.19	Interrupt Mondo Data 0 Register (0x0062C000, 0x0072C000 / 0x0) 210	
1.3.4.20	Interrupt Mondo Data 1 Register (0x0062C008, 0x0072C008 / 0x0) 210	
1.3.4.21	ERR COR Mapping Register (0x00630000, 0x00730000 / 0x0)	211
1.3.4.22	ERR NONFATAL Mapping Register (0x00630008, 0x00730008 / 0x0)	211
1.3.4.23	ERR FATAL Mapping Register (0x00630010, 0x00730010 / 0x0) 212	
1.3.4.24	PM PME Mapping Register (0x00630018, 0x00730018 / 0x0)	212
1.3.4.25	PME To ACK Mapping Register (0x00630020, 0x00730020 / 0x0) 213	
1.3.4.26	IMU Error Log Enable Register (0x00631000, 0x00731000 / 0x7FFF)	213
1.3.4.27	IMU Interrupt Enable Register (0x00631008, 0x00731008 / 0x0) 214	
1.3.4.28	IMU Interrupt Status Register (0x00631010, 0x00731010 / 0x0)	216
1.3.4.29	IMU Error Status Clear Register (0x00631018, 0x00731018 / 0x0) 218	

1.3.4.30	IMU Error Status Set Register (0x00631020, 0x00731020 / 0x0)	220
1.3.4.31	IMU RDS Error Log Register (0x00631028, 0x00731028 / 0x0)	222
1.3.4.32	IMU SCS Error Log Register (0x00631030, 0x00731030 / 0x0)	224
1.3.4.33	IMU EQS Error Log Register (0x00631038, 0x00731038 / 0x0)	225
1.3.4.34	DMC Core and Block Interrupt Enable Register (0x00631800, 0x00731800 / 0x0)	225
1.3.4.35	DMC Core and Block Error Status Register (0x00631808, 0x00731808 / 0x0)	226
1.3.4.36	Multi Core Error Status Register (0x00631810, 0x00731810 / 0x0)	226
1.3.4.37	IMU Performance Counter Select Register (0x00632000, 0x00732000 / 0x0)	227
1.3.4.38	IMU Performance Counter Zero Register (0x00632008, 0x00732008 / 0x0)	227
1.3.4.39	IMU Performance Counter One Register (0x00632010, 0x00732010 / 0x0)	228
1.3.4.40	MSI 32-bit Address Register (0x00634000, 0x00734000 / 0x0)	228
1.3.4.41	MSI 64-bit Address Register (0x00634008, 0x00734008 / 0x0)	228
1.3.4.42	Mem 64 PCIE Offset Register (0x00634018, 0x00734018 / 0x0)	229
1.3.4.43	MMU Control and Status Register (0x00640000, 0x00740000 / 0x0)	230
1.3.4.44	MMU TSB Control Register (0x00640008, 0x00740008 / 0x0)	231
1.3.4.45	MMU TTE Cache Flush Address Register (0x00640100, 0x00740100 / 0x0)	232
1.3.4.46	MMU TTE Cache Invalidate Register (0x00640108, 0x00740108 / 0x0)	233
1.3.4.47	MMU Error Log Enable Register (0x00641000, 0x00741000 / 0xFFFF)	233
1.3.4.48	MMU Interrupt Enable Register (0x00641008, 0x00741008 / 0x0)	233
1.3.4.49	MMU Interrupt Status Register (0x00641010, 0x00741010 / 0x0)	234
1.3.4.50	MMU Error Status Clear Register (0x00641018, 0x00741018 / 0x0)	234
1.3.4.51	MMU Error Status Set Register (0x00641020, 0x00741020 / 0x0)	236



1.3.4.52	MMU Translation Fault Address Register (0x00641028, 0x00741028 / 0x0)	238
1.3.4.53	MMU Translation Fault Status Register (0x00641030, 0x00741030 / 0x0)	239
1.3.4.54	MMU Performance Counter Select Register (0x00642000, 0x00742000 / 0x0)	239
1.3.4.55	MMU Performance Counter Zero Register (0x00642008, 0x00742008 / 0x0)	240
1.3.4.56	MMU Performance Counter One Register (0x00642010, 0x00742010 / 0x0)	240
1.3.4.57	MMU TTE Cache Virtual Tag Registers (0x00646000-0x006461F8, 0x00746000-0x007461F8 / 0x0)	240
1.3.4.58	MMU TTE Cache Physical Tag Registers (0x00647000-0x006471F8, 0x00747000-0x007471F8 / 0x0)	241
1.3.4.59	MMU TTE Cache Data Registers (0x00648000-0x00648FF8, 0x00748000-0x00748FF8 / 0x0)	241
1.3.4.60	ILU Error Log Enable Register (0x00651000, 0x00751000 / 0xF0) 242	
1.3.4.61	ILU Interrupt Enable Register (0x00651008, 0x00751008 / 0x0) 242	
1.3.4.62	ILU Interrupt Status Register (0x00651010, 0x00751010 / 0x0) 243	
1.3.4.63	ILU Error Status Clear Register (0x00651018, 0x00751018 / 0x0) 243	
1.3.4.64	ILU Error Status Set Register (0x00651020, 0x00751020 / 0x0) 244	
1.3.4.65	PEC Core and Block Interrupt Enable Register (0x00651800, 0x00751800 / 0x0)	245
1.3.4.66	PEC Core and Block Interrupt Status Register (0x00651808, 0x00751808 / 0x0)	246
1.3.4.67	ILU Device Capabilities Register (0x00652000, 0x00752000 / 0x0) 246	
1.3.4.68	DMC Debug Select Register for Port A (0x00653000, 0x00753000 / 0x0)	246
1.3.4.69	DMC Debug Select Register for Port B (0x00653008, 0x00753008 / 0x0)	247
1.3.4.70	DMC PCI Express Configuration Register (0x00653100, 0x00753100 / 0x0)	248
1.3.4.71	Packet Scoreboard DMA Register Set (0x00660000-0x006600F8, 0x00760000-0x007600F8 / 0x0)	249

1.3.4.72	Packet Scoreboard PIO Register Set (0x00664000-0x00664078, 0x00764000-0x00764078 / 0x0)	249
1.3.4.73	Transaction Scoreboard Register Set (0x00670000-0x006700F8, 0x00770000-0x007700F8 / 0x0)	250
1.3.4.74	Transaction Scoreboard Status Register (0x00670100, 0x00770100 / 0x1)	250
1.3.4.75	TLU Control Register (0x00680000, 0x00780000 / 0x101) ...	251
1.3.4.76	TLU Status Register (0x00680008, 0x00780008 / 0x0)	253
1.3.4.77	TLU PME Turn Off Generate Register (0x00680010, 0x00780010 / 0x0)	253
1.3.4.78	TLU Ingress Credits Initial Register (0x00680018, 0x00780018 / 0x10000200C0)	253
1.3.4.79	TLU Diagnostic Register (0x00680100, 0x00780100 / 0x0) ...	254
1.3.4.80	TLU Egress Credits Consumed Register (0x00680200, 0x00780200 / 0x0)	257
1.3.4.81	TLU Egress Credit Limit Register (0x00680208, 0x00780208 / 0x0) 257	
1.3.4.82	TLU Egress Retry Buffer Register (0x00680210, 0x00780210 / 0x0) 258	
1.3.4.83	TLU Ingress Credits Allocated Register (0x00680218, 0x00780218 / 0x0)	258
1.3.4.84	TLU Ingress Credits Received Register (0x00680220, 0x00780220 / 0x0)	259
1.3.4.85	TLU Other Event Log Enable Register (0x00681000, 0x00781000 / 0xFFFFFFFF)	259
1.3.4.86	TLU Other Event Interrupt Enable Register (0x00681008, 0x00781008 / 0x0)	260
1.3.4.87	TLU Other Event Interrupt Status Register (0x00681010, 0x00781010 / 0x0)	260
1.3.4.88	TLU Other Event Status Clear Register (0x00681018, 0x00781018 / 0x0)	260
1.3.4.89	TLU Other Event Status Set Register (0x00681020, 0x00781020 / 0x0)	264
1.3.4.90	TLU Receive Other Event Header1 Log Register (0x00681028, 0x00781028 / 0x0)	267
1.3.4.91	TLU Receive Other Event Header2 Log Register (0x00681030, 0x00781030 / 0x0)	268



1.3.4.92	TLU Transmit Other Event Header1 Log Register (0x00681038, 0x00781038 / 0x0)	268
1.3.4.93	TLU Transmit Other Event Header2 Log Register (0x00681040, 0x00781040 / 0x0)	269
1.3.4.94	TLU Performance Counter Select Register (0x00682000, 0x00782000 / 0x0)	270
1.3.4.95	TLU Performance Counter Zero Register (0x00682008, 0x00782008 / 0x0)	272
1.3.4.96	TLU Performance Counter One Register (0x00682010, 0x00782010 / 0x0)	272
1.3.4.97	TLU Performance Counter Two Register (0x00682018, 0x00782018 / 0x0)	272
1.3.4.98	TLU Debug Select A Register (0x00683000, 0x00783000 / 0x0)	272
1.3.4.99	TLU Debug Select B Register (0x00683008, 0x00783008 / 0x0)	273
1.3.4.100	TLU Device Capabilities Register (0x00690000, 0x00790000 / 0x2) 274	
1.3.4.101	TLU Device Control Register (0x00690008, 0x00790008 / 0x0)	274
1.3.4.102	TLU Device Status Register (0x00690010, 0x00790010 / 0x0)	275
1.3.4.103	TLU Link Capabilities Register (0x00690018, 0x00790018 / 0x15C81).....	276
1.3.4.104	TLU Link Control Register (0x00690020, 0x00790020 / 0x0) .	276
1.3.4.105	TLU Link Status Register (0x00690028, 0x00790028 / 0x0) ..	278
1.3.4.106	TLU Slot Capabilities Register (0x00690030, 0x00790030 / 0x0) 279	
1.3.4.107	TLU Uncorrectable Error Log Enable Register (0x00691000, 0x00791000 / 0x17F011)	279
1.3.4.108	TLU Uncorrectable Error Interrupt Enable Register (0x00691008, 0x00791008 / 0x0)	280
1.3.4.109	TLU Uncorrectable Error Interrupt Status Register (0x00691010, 0x00791010 / 0x0)	280
1.3.4.110	TLU Uncorrectable Error Status Clear Register (0x00691018, 0x00791018 / 0x0)	280
1.3.4.111	TLU Uncorrectable Error Status Set Register (0x00691020, 0x00791020 / 0x0)	282
1.3.4.112	TLU Receive Uncorrectable Error Header1 Log Register (0x00691028, 0x00791028 / 0x0)	284

1.3.4.113	TLU Receive Uncorrectable Error Header2 Log Register (0x00691030, 0x00791030 / 0x0)	284
1.3.4.114	TLU Transmit Uncorrectable Error Header1 Log Register (0x00691038, 0x00791038 / 0x0)	285
1.3.4.115	TLU Transmit Uncorrectable Error Header2 Log Register (0x00691040, 0x00791040 / 0x0)	285
1.3.4.116	TLU Correctable Error Log Enable Register (0x006A1000, 0x007A1000 / 0x11C1)	286
1.3.4.117	TLU Correctable Error Interrupt Enable Register (0x006A1008, 0x007A1008 / 0x0)	286
1.3.4.118	TLU Correctable Error Interrupt Status Register (0x006A1010, 0x007A1010 / 0x0)	286
1.3.4.119	TLU Correctable Error Status Clear Register (0x006A1018, 0x007A1018 / 0x0)	287
1.3.4.120	TLU Correctable Error Status Set Register (0x006A1020, 0x007A1020 / 0x0)	288
1.3.4.121	LPU ID Register (0x006E2000, 0x007E2000 / 0x0)	288
1.3.4.122	LPU Reset Register (0x006E2008, 0x007E2008 / 0x0)	290
1.3.4.123	LPU Debug Status Register (0x006E2010, 0x007E2010 / 0x0)	291
1.3.4.124	LPU Debug Config Register (0x006E2018, 0x007E2018 / 0x0)	292
1.3.4.125	LPU LTSSM Control Register (0x006E2020, 0x007E2020 / 0x0)	292
1.3.4.126	LPU Link Status Register (0x006E2028, 0x007E2028 / 0x1) .	295
1.3.4.127	LPU Interrupt Status Register (0x006E2040, 0x007E2040 / 0x0)	296
1.3.4.128	LPU Interrupt Mask Register (0x006E2048, 0x007E2048 / 0x800000FF)	297
1.3.4.129	LPU Link Performance Counter Select Register (0x006E2100, 0x007E2100 / 0x0)	299
1.3.4.130	LPU Link Performance Counter Control Register (0x006E2110, 0x007E2110 / 0x0)	300
1.3.4.131	LPU Link Performance Counter1 (0x006E2120, 0x007E2120 / 0x0)	301
1.3.4.132	LPU Link Performance Counter1 Test (0x006E2128, 0x007E2128 / 0x0)	301
1.3.4.133	LPU Link Performance Counter2 (0x006E2130, 0x007E2130 / 0x0)	302



1.3.4.134	LPU Link Performance Counter2 Test (0x006E2138, 0x007E2138 / 0x0)	302
1.3.4.135	LPU Link Layer Config Register (0x006E2200, 0x007E2200 / 0x100)	302
1.3.4.136	LPU Link Layer Status Register (0x006E2208, 0x007E2208 / 0x1) 305	
1.3.4.137	LPU Link Layer Interrupt and Status Register (0x006E2210, 0x007E2210 / 0x0)	306
1.3.4.138	LPU Link Layer Interrupt and Status Test Register (0x006E2218, 0x007E2218 / 0x0)	307
1.3.4.139	LPU Link Layer Interrupt Mask Register (0x006E2220, 0x007E2220 / 0x807FFFFFFF)	308
1.3.4.140	LPU Flow Control Update Control Register (0x006E2240, 0x007E2240 / 0x3)	309
1.3.4.141	LPU Link Layer Flow Control Update Timeout Value Register (0x006E2260, 0x007E2260 / 0x1D4C)	310
1.3.4.142	LPU Link Layer VC0 Flow Control Update Timer0 Register (0x006E2268, 0x007E2268 / 0x0)	310
1.3.4.143	LPU Link Layer VC0 Flow Control Update Timer1 Register (0x006E2270, 0x007E2270 / 0x0)	311
1.3.4.144	LPU Txlink Frequent Nak Latency Timer Threshold Register (0x006E2400, 0x007E2400 / 0x43)	311
1.3.4.145	LPU Txlink AckNak Latency Timer Register (0x006E2408, 0x007E2408 / 0x0)	312
1.3.4.146	LPU Txlink Replay Timer Threshold Register (0x006E2410, 0x007E2410 / 0xFC)	312
1.3.4.147	LPU Txlink Replay Timer Register (0x006E2418, 0x007E2418 / 0xFC)	312
1.3.4.148	LPU Txlink Replay Number Status Register (0x006E2420, 0x007E2420 / 0x0)	313
1.3.4.149	LPU Replay Buffer Max Address Register (0x006E2428, 0x007E2428 / 0xB3F)	313
1.3.4.150	LPU Txlink Retry FIFO Pointer Register (0x006E2430, 0x007E2430 / 0xFFFF0000)	314
1.3.4.151	LPU Txlink Retry FIFO R/W Pointer Register (0x006E2438, 0x007E2438 / 0x0)	314
1.3.4.152	LPU Txlink Retry FIFO Credit Register (0x006E2440, 0x007E2440 / 0xB40)	315

1.3.4.153	LPU Txlink Sequence Counter Register (0x006E2448, 0x007E2448 / 0xFFF0000)	315
1.3.4.154	LPU Txlink Ack Sent Sequence Number Register (0x006E2450, 0x007E2450 / 0xFFF)	316
1.3.4.155	LPU Txlink Sequence Count FIFO Max Addr Register (0x006E2458, 0x007E2458 / 0x167)	316
1.3.4.156	LPU Txlink Sequence Count FIFO Pointers Register (0x006E2460, 0x007E2460 / 0xFFF0000).....	316
1.3.4.157	LPU Txlink Sequence Count R/W Pointers Register (0x006E2468, 0x007E2468 / 0x0).....	317
1.3.4.158	LPU Txlink Test Control Register (0x006E2470, 0x007E2470 / 0x0)	317
1.3.4.159	LPU Txlink Memory Address Control Register (0x006E2480, 0x007E2480 / 0x0).....	318
1.3.4.160	LPU Txlink Memory Data Load0 Register (0x006E2488, 0x007E2488 / 0x0).....	319
1.3.4.161	LPU Txlink Memory Data Load1 Register (0x006E2490, 0x007E2490 / 0x0).....	319
1.3.4.162	LPU Txlink Memory Data Load2 Register (0x006E2498, 0x007E2498 / 0x0).....	320
1.3.4.163	LPU Txlink Memory Data Load3 Register (0x006E24A0, 0x007E24A0 / 0x0)	320
1.3.4.164	LPU Txlink Memory Data Load4 Register (0x006E24A8, 0x007E24A8 / 0x0)	320
1.3.4.165	LPU Txlink Retry Data Count Register (0x006E24C0, 0x007E24C0 / 0x0)	321
1.3.4.166	LPU Txlink Sequence Buffer Count Register (0x006E24C8, 0x007E24C8 / 0x0)	321
1.3.4.167	LPU Txlink Sequence Buffer Bottom Data Register (0x006E24D0, 0x007E24D0 / 0x0)	321
1.3.4.168	LPU Txlink Ack Latency Timer Threshold Register (0x006E24E0, 0x007E24E0 / 0x5).....	322
1.3.4.169	LPU Rxlink Next Receive Sequence + 1 Counter Register (0x006E2500, 0x007E2500 / 0x1)	322
1.3.4.170	LPU Rxlink Unsupported DLLP Received Register (0x006E2508, 0x007E2508 / 0x0).....	323
1.3.4.171	LPU Rxlink Test Control Register (0x006E2510, 0x007E2510 / 0x0)	323



1.3.4.172	LPU Physical Layer Configuration Register (0x006E2600, 0x007E2600 / 0x10).....	323
1.3.4.173	LPU Phy Layer Status Register (0x006E2608, 0x007E2608 / 0x0) 326	
1.3.4.174	LPU Phy Layer Interrupt and Status Register (0x006E2610, 0x007E2610 / 0x0).....	326
1.3.4.175	LPU Phy Interrupt and Status Test Register (0x006E2618, 0x007E2618 / 0x0).....	327
1.3.4.176	LPU Phy Interrupt Mask Register (0x006E2620, 0x007E2620 / 0x80000FFF).....	328
1.3.4.177	LPU Receive Phy Config Register (0x006E2680, 0x007E2680 / 0x0) 329	
1.3.4.178	LPU Receive Phy Status1 Register (0x006E2688, 0x007E2688 / 0x0).....	331
1.3.4.179	LPU Receive Phy Status2 Register (0x006E2690, 0x007E2690 / 0x0).....	331
1.3.4.180	LPU Receive Phy Status3 Register (0x006E2698, 0x007E2698 / 0x0).....	332
1.3.4.181	LPU Receive Phy Interrupt and Status Register (0x006E26A0, 0x007E26A0 / 0x0).....	333
1.3.4.182	LPU Receive Phy Interrupt and Status Test Register (0x006E26A8, 0x007E26A8 / 0x0).....	333
1.3.4.183	LPU Receive Phy Interrupt Mask Register (0x006E26B0, 0x007E26B0 / 0x80000FFF).....	334
1.3.4.184	LPU Transmit Phy Config Register (0x006E2700, 0x007E2700 / 0x0).....	334
1.3.4.185	LPU Transmit Phy Status Register (0x006E2708, 0x007E2708 / 0x73000010).....	335
1.3.4.186	LPU Transmit Phy Interrupt and Status Register (0x006E2710, 0x007E2710 / 0x0).....	336
1.3.4.187	LPU Transmit Phy Interrupt and Status Test Register (0x006E2718, 0x007E2718 / 0x0).....	337
1.3.4.188	LPU Transmit Phy Interrupt Mask Register (0x006E2720, 0x007E2720 / 0x80000FFF).....	337
1.3.4.189	LPU Transmit Phy Status 2 Register (0x006E2728, 0x007E2728 / 0x0).....	338
1.3.4.190	LPU LTSSM Config1 Register (0x006E2780, 0x007E2780 / 0x1905) 339	

1.3.4.191	LPU LTSSM Config2 Register (0x006E2788, 0x007E2788 / 0x2DC6C0).....	340
1.3.4.192	LPU LTSSM Config3 Register (0x006E2790, 0x007E2790 / 0x7A120)	341
1.3.4.193	LPU LTSSM Config4 Register (0x006E2798, 0x007E2798 / 0x29C00).....	341
1.3.4.194	LPU LTSSM Config5 Register (0x006E27A0, 0x007E27A0 / 0x0) 342	
1.3.4.195	LPU LTSSM Status1 Register (0x006E27A8, 0x007E27A8 / 0x0) 343	
1.3.4.196	LPU LTSSM Status2 Register (0x006E27B0, 0x007E27B0 / 0x0) 346	
1.3.4.197	LPU LTSSM Interrupt and Status Register (0x006E27B8, 0x007E27B8 / 0x0).....	346
1.3.4.198	LPU LTSSM Interrupt and Status Test Register (0x006E27C0, 0x007E27C0 / 0x0)	348
1.3.4.199	LPU LTSSM Interrupt Mask Register (0x006E27C8, 0x007E27C8 / 0x8000FFFF).....	349
1.3.4.200	LPU LTSSM Status Write Enable Register (0x006E27D0, 0x007E27D0 / 0x0)	350
1.3.4.201	LPU SERDES Glue Config1 Register (0x006E2800, 0x007E2800 / 0x89019)	351
1.3.4.202	LPU SERDES Glue Config2 Register (0x006E2808, 0x007E2808 / 0xA16BF1B5)	353
1.3.4.203	LPU SERDES Glue Config3 Register (0x006E2810, 0x007E2810 / 0x4401F4)	354
1.3.4.204	LPU SERDES Glue Config4 Register (0x006E2818, 0x007E2818 / 0x1E848).....	356
1.3.4.205	LPU SERDES Glue Status Register (0x006E2820, 0x007E2820 / 0xFFFF0000).....	356
1.3.4.206	LPU SERDES Glue Interrupt and Status Register (0x006E2828, 0x007E2828 / 0x0).....	357
1.3.4.207	LPU SERDES Glue Interrupt and Status Test Register (0x006E2830, 0x007E2830 / 0x0)	358
1.3.4.208	LPU SERDES Glue Interrupt Mask Register (0x006E2838, 0x007E2838 / 0x80FFFFFF)	358
1.3.4.209	LPU SERDES Glue Power Down1 Register (0x006E2840, 0x007E2840 / 0x0).....	359



1.3.4.210	LPU SERDES Glue Power Down2 Register (0x006E2848, 0x007E2848 / 0x0)	360
1.3.4.211	LPU SERDES Glue Config5 Register (0x006E2850, 0x007E2850 / 0x0)	362
1.4	Operational Sequences	363
1.4.1	Power-Up Reset (Hard Reset)	363
1.4.1.1	Software-Initiated Hard Reset	363
1.4.2	Power-On Reset (Soft Reset)	363
1.4.2.1	Software-Initiated Soft Reset	364
1.4.2.2	PCI Express Link Disable	364
1.4.2.3	PCI Express Hot Reset	364
1.4.3	Externally Initiated Reset (XIR)	365
1.4.3.1	Soft XIR	365
1.4.3.2	Button/Watchdog XIR	365
1.4.4	Reset Summary	365
1.4.5	POR & Warm Reset Initialization	366
1.4.5.1	Example Address Initialization	366
1.4.5.2	Example MMU Initialization	367
1.4.5.3	Example PCI-E Link Initialization	368
1.4.5.4	Example Interrupt Mapping Initialization	368
1.4.5.5	Link Training	368
1.4.5.6	PCI Express Configuration	369
1.4.6	Estar Sequence	369
1.4.7	Internal Loopback Program Instructions	370
1.4.8	The Drain State	371
1.4.9	Re-train a Link After it is Down	372
1.4.10	PCI-E Device Power Off and On	372
1.4.11	SERDES Electrical SERDES Configuration	373
1.4.12	DR	374
1.4.12.1	Remapping of functionality	374
1.4.12.2	Removal from active operation	374
1.4.12.3	Interrupt Redirection	374

1.4.13	Error Event Summary	376
1.4.14	Interrupt Processing for Mondos 62 and 63	399
1.4.14.1	Fire Internal JBC Error Mondo 63.....	399
1.4.14.2	Fire Internal DMC & PEC Error or Event Mondo 62	401
1.5	Software Visible State	408
1.5.1	Accessing BIST Results	408
1.6	Fire Error Injection	408
1.6.1	JBus Error Injection.....	408
1.6.1.1	MMU Data Cache Error Injection.....	409
1.6.1.2	PCI-E Interface Error Injection/Force	409



1.1 Introduction

1.1.1 Chapter Organization

The programmer's reference manual consists of the following chapters:

- Introduction
- Operational Overview
- CSR Fields and Bits
- Operational Sequences
- Software Visible State
- Fire Error Injection

The Introduction chapter gives an overview of this document and a high level overview of Fire from a software perspective.

The Operational Overview chapter provides a detailed description of the functionality within Fire.

The CSR Fields and Bits chapter defines and documents the registers within Fire.

The Operational Sequences chapter provides software sequences/algorithms for:

- Chip initialization/configuration
- Common sequences during normal operation
- Interrupt handling
- Error handling

The Software Visible State chapter describes any information that can be viewed via software interface beyond the base CSR set (e.g. units which are only accessible through scan interfaces).

The Fire Error Injection chapter provides detail on how errors can be injected either internally within Fire, or on one of the external buses (JBus or PCIE).

1.1.2 Fire Overview

Fire is a companion core-logic ASIC to JBus based 64-bit SPARC V9 CPUs. JBus is a split transaction 16-byte shared address/data bus. The central task of Fire is to be the point of access to I/O, graphics, and system interrupts for a uniprocessor up to an 8-way (via CMP processors) symmetric multi-processor system. For those familiar with PC system hardware partitioning, Fire is the equivalent of an PC Northbridge, minus the interface to main-memory. The interface to main-memory comes directly off the CPU in our case.

One or two Fires can be present in a system. Fire can be configured as the master I/O device (e.g. will handle boot) or the slave I/O device via an external pin.

PCI Express ports are point to point dual simplex interfaces. A x1 (pronounced "by 1") PCI Express port provides a low voltage differential signal pair (one in each direction) running at 2.5 Gbit/s (8b10b). The maximum theoretical bandwidth for a x1 is 2 GBit/s in each direction or 512 MBytes/s aggregate. A x4 PCI Express port provides four differential signal pairs which produces a maximum theoretical aggregate bandwidth of 2 GBytes/s (1 GByte/s in each direction).

Fire provides two x8 PCI Express ports (1.0a compliant) with a maximum theoretical aggregate bandwidth of 4 GBytes/s (2 GBytes/s in each direction). JBus provides a maximum theoretical bandwidth of approximately 2.5 GBytes/s. The bandwidth to system memory (from JBus) though the CPU's memory controller can be less than the JBus bandwidth (e.g. Jalapeno's memory controller's maximum bandwidth for JBus accesses to local memory is ~ 1.7 GBytes/s). Therefore, Fire's performance will be limited by either JBus bandwidth, or memory bandwidth.

Fire supports x1, x4, and x8 PCI Express ports in two connections.

Fire provides the following physical interfaces:

- Single JBus port (uses two AgentIDs)
- Two x8 PCI Express Ports
- Internal Interrupt Concentrator (supports up to 40 external interrupts)
- Slave only (no DMA) E-bus interface (for boot support)
- Eight GPIO pins
- Two i2c master-capable ports

Fire provides the following high level functionality:

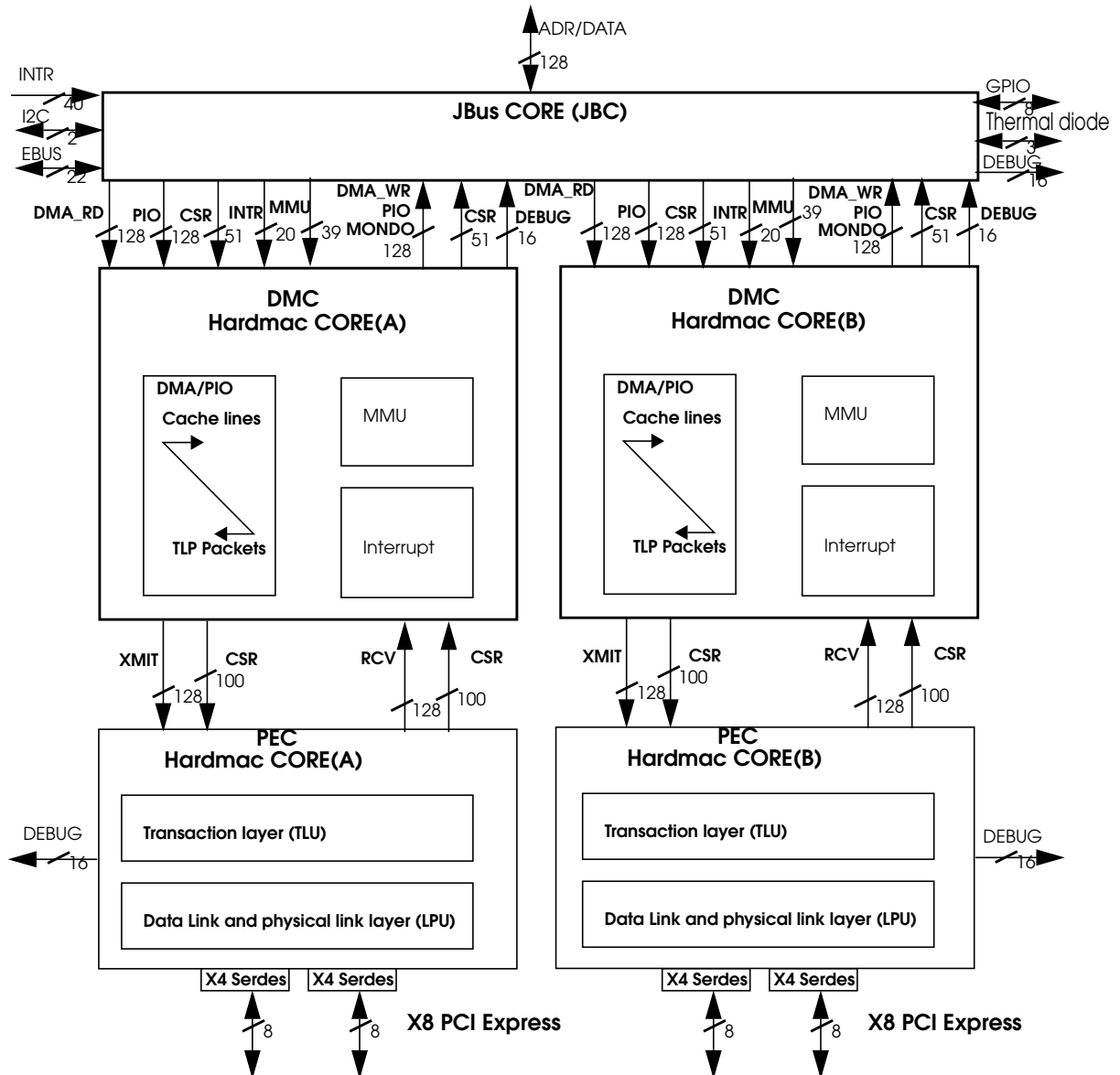
- Mapping Ebus address space, PCI Express configuration, PCI Express I/O, and PCI Express 32-bit and 64-bit memory address spaces into JBus address space (downbound operation).
- Mapping JBus address space into PCI Express memory address space (upbound operation). Both bypass and translated accesses through an MMU are supported.
- Translating PCI Express MSI interrupts into JBus interrupts.
- PCI Express legacy INTx support.
- PCI Express Power Management support (excluding wake on PME).
- Integrated interrupt concentrator (supports up to 40 external interrupts)

Fire does not support the following PCI Express features:

- Peer to peer communication (i.e. between two JBus I/O devices)
- Isochronous transfers (only one virtual channel is supported)
- Lock Operations
- Wake System from PME (i.e. no Vaux support)

1.1.3 Fire Block Diagram

Figure 1-1 Fire Block Diagram



1.2 Operational Overview

1.2.1 Fire Overview

Fire has a JBus interface to the CPUs & memory, a slave-only Ebus leaf which supports booting, eight GPIO pins, two i2c master controllers, and two independent PCI Express leaves.

Each PCI Express leaf contains an MMU, a TSB cache, 36 Event Queues, MSI mapping state, an interrupt concentrator to support non PCI Express interrupts, and a JBus Interrupt Mondo Controller.

Fire supports forty external interrupts. Twenty external interrupts are dedicated to the PCIE-A leaf and twenty external interrupts are dedicated to the PCIE-B leaf.

On PCI Express, the requester ID is the combination of a Requester's Bus Number, Device Number, and Function Number that uniquely identifies the Requester. Fire's PCI Express requester ID is programmable by software through "DMC PCI Express Configuration" Register.

1.2.2 JBus to Fire (Downbound) Transactions

1.2.2.1 JBUS Physical Address Partitioning

For JBus, the 43-bit physical address space is partitioned such that each JBus agent (device attached to JBus) has the following regions defined based on its JBus AgentID.

- 8 MByte noncacheable configuration region
- 64 GByte noncacheable region
- 64 GByte cacheable region

The base address of the 8 Mbyte noncacheable configuration region is {1,0,13'b0,AgentID[4:0], 23'b0}.

Table 1-1 8 MByte Noncacheable Configuration Region Addressing

bits	Description
42 (1)	Set to 1 to indicate noncacheable space
41 (1)	Set to 0 to indicate 8 MByte configuration region within the noncacheable space
40:28 (13)	Set to zero.
27:23 (5)	The JBus AgentID that the device occupies.
22:00 (23)	Offset within the 8 MByte noncacheable region

The base address of the 64 GByte noncacheable region is {1,1,AgentID[4:0], 36'b0}.

Table 1-2 64 GByte Noncacheable Region Addressing

bits	Description
42 (1)	Set to 1 to indicate noncacheable space
41 (1)	Set to 1 to indicate 64 GByte region within the noncacheable space
40:36 (5)	The JBus AgentID that the device occupies
35:00 (36)	Offset within the 64 GByte noncacheable region

The base address of the 64 GByte cacheable region is {0,0,AgentID[4:0], 36'b0}.

Table 1-3 64 GByte Cacheable Region Addressing

bits	Description
42 (1)	Set to 0 to indicate cacheable space (causes unmapped error in Fire, see below)
41 (1)	From JBus spec: Set to 0 for normal cacheable space. A setting of 1 indicates an undefined address space. Devices may ignore this bit when decoding cacheable addresses, effectively creating an alias, but such aliases are not cache coherent as far as Fire is concerned.
40:36 (5)	The JBus AgentID that the device occupies
35:00 (36)	Offset within the 64 GByte cacheable region

Programming Note:

The JBus spec is unclear as to what the actual requirements are around the use of bit 41. Fire **DOES** use bit 41 in its snoop compare logic. Two addresses that differ only in bit 41 are **NOT** treated as aliases by Fire. This functionality matches that of Tomatillo.

Fire uses two AgentIDs to allow it to generate more outstanding transactions on JBus. The JBus Address regions associated with the two AgentIDs alias to the same address regions within Fire (e.g. the two 8 MByte noncacheable configuration regions will map to the same registers within Fire).

Fire uses the 8 MByte noncacheable configuration region and the 64 GByte noncacheable region. Fire does not use the 64 GByte cacheable region, although it still responds to reads and writes to its cacheable space in order to avoid system hangs. Fire returns a Unmapped Read Error Packet on JBus for reads to cacheable space. The error is logged as a PIO Unmapped Read Error (PIO_UNMAP_RD) Error in the JBC Error Status Clear Register. Fire drops writes to cacheable space and logs the event as a JBus Unmapped Error (JUE) in Fire's JBC Error Status Clear Register.

The 8 MByte noncacheable configuration region contains all of Fire's registers.

- A partition of the 8 MByte noncacheable configuration region can be found in Table 1-7, "8 MByte Noncacheable Configuration Region," on page 28.
- Writes to an address in a reserved range in Fire's 8MByte noncacheable configuration region are dropped and the event is logged as a PIO Unmapped Error (PIO_UNMAP) in the JBC Error Status Clear Register.

- Reads to an address in a reserved range in Fire's 8MByte noncacheable configuration region will return an Unmapped Read Error Packet on JBus. The event is logged as a PIO Unmapped Read Error (PIO_UNMAP_RD) in the JBC Error Status Clear Register.
- A map of the registers within the configuration region can be found in Table 1-19, "Fire Register Map," on page 60.
- Writes to an address marked as RESERVED in Table 1-19 on page 60 are silently dropped. No error is logged.
- Reads to an address marked as RESERVED in Table 1-19 on page 60 (except the 8k space dedicated to LPU CSRs on both PCIE-A and PCIE-B sides) will return an Unmapped Read Error Packet on JBus. No error is logged.
- Reads to an address marked as RESERVED in the 8k LPU CSR's space will return zero's. No error is logged. The 8k LPU CSR's space on PCIE-A side is 0x6E2000 - 0x6E4000 and on PCIE-B side is 0x7E2000 - 0x7E4000.
- All accesses to the 8 MByte noncacheable configuration region are big endian. Noncacheable block reads and block writes are not allowed to this region.
- Block writes are dropped and the error is logged as a JBus Unmapped Error (JUE) Error in the JBC Error Status Clear Register.
- Block reads will return an Unmapped Read Error Packet on JBus. The error is logged as a PIO Unmapped Read Error (PIO_UNMAP_RD) Error in the JBC Error Status Clear Register.

Accesses to Fire's 64 GByte noncacheable region **outside** the sub-regions mapped by the Address Mask/Match registers (described below) are treated as follows:

- Reads will return an Unmapped Read Error Packet on JBus. The error is logged as a PIO Unmapped Read Error (PIO_UNMAP_RD) in the JBC Error Status Clear Register.
- Writes are dropped. The error is logged as a PIO Unmapped Error (PIO_UNMAP) in the JBC Error Status Clear Register.

The 64 GByte noncacheable space is further broken up into multiple sub-regions. Fire decodes and responds to 7 separate sub-regions. There is one sub-region for the Ebus leaf and three sub-regions for each of the two PCI Express leaves. Each sub-region's offset and size are programmed during chip initialization via offset base and offset mask registers (Section 1.2.2.2, "Offset Base and Offset Mask Register Behavior," on page 1-27).

The three sub-regions in a PCI Express leaf are the PCI Express configuration & IO sub-region, the 32-bit addressable PCI Express memory sub-region, and the 64-bit addressable PCI Express memory sub-region.

The sub-regions within the 64 Gbyte noncacheable region are summarized in Table 1-4, “64 GByte Noncacheable Region Partitioning,” on page 26.

Table 1-4 64 GByte Noncacheable Region Partitioning

Sub-region	Size	JBus to Fire Transactions
Ebus	128Mb	noncacheable read (16 byte max) (NCRD) noncacheable write (16 byte max) (NCWR & NCWRC)
PCIE-A CFG/IO	512Mb	noncacheable read (4 byte max) (NCRD) noncacheable write (4 byte max) (NCWR & NCWRC)
PCIE-A Mem32	16MB - 2GB	noncacheable read (NCRD) noncacheable write (NCWR & NCWRC) noncacheable block read (NCBRD) noncacheable block write (NCBWR)
PCIE-A Mem64	16MB - 32GB	noncacheable read (NCRD) noncacheable write (NCWR & NCWRC) noncacheable block read (NCBRD) noncacheable block write (NCBWR)
PCIE-B CFG/IO	512Mb	noncacheable read (4 byte max) (NCRD) noncacheable write (4 byte max) (NCWR & NCWRC)
PCIE-B Mem32	16MB - 2GB	noncacheable read (NCRD) noncacheable write (NCWR & NCWRC) noncacheable block read (NCBRD) noncacheable block write (NCBWR)
PCIE-B Mem64	16MB - 32GB	noncacheable read (NCRD) noncacheable write (NCWR & NCWRC) noncacheable block read (NCBRD) noncacheable block write (NCBWR)

Only certain size accesses are allowed to the different 64 G and 8 M regions. Table 1-5 on page 26 lists the allowable accesses for NCWR transactions and Table 1-6 on page 27 lists the allowable accesses for NCRD transactions. Note that NCWR’s and NCRD’s are handled differently. NCBWR and NCBRD transactions are only allowed to access Mem 32 and Mem 64 space.

Table 1-5 Allowable NCWR access sizes (Sheet 1 of 2)

Size in Bytes	Conf/IO	64 G space			8 M space	
		Mem32	Mem 64	Ebus	CSR	
1	Y	Y	Y	Y	N	
2	Y	Y	Y	Y	N	
3	Y	Y	Y	N	N	
4	Y	Y	Y	Y	N	
5	N	Y	Y	N	N	

Table 1-5 Allowable NCWR access sizes (Sheet 2 of 2)

Size in Bytes	Conf/IO	64 G space		8 M space	
		Mem32	Mem 64	Ebus	CSR
6	N	Y	Y	N	N
7	N	Y	Y	N	N
8	N	Y	Y	Y	Y
16	N	Y	Y	Y	N

Table 1-6 Allowable NCRD access sizes

Size in Bytes	Conf/IO	64 G space		8 M space	
		Mem32	Mem 64	Ebus	CSR
1	Y	Y	Y	Y	N
2	Y	Y	Y	Y	N
3	N	N	N	N	N
4	Y	Y	Y	Y	N
5	N	N	N	N	N
6	N	N	N	N	N
7	N	N	N	N	N
8	N	Y	Y	Y	Y
16	N	Y	Y	Y	N

1.2.2.2 Offset Base and Offset Mask Register Behavior

Fire is using a common implementation for all of its address match and mask registers. In some cases this leads to extra bits that are not strictly necessary as they allow Fire to map a sub-region larger than what is implemented by the corresponding sub-region. Fire uses these registers to determine where the various sub-regions live within the 64 GByte noncacheable region.

Fire first checks PA bits [42:36] ([42:41] == 11, [40:36] == AgentID[*]) to see if it should respond to the JBus Address, where * = one of the 2 possible Agent ID's for the given slot which Fire is in (1C or 1D for slot 4 or 1E and 1F for slot 5). If the address is found to be within its memory range, bits [35:24] are masked with (bitwise AND) Offset Mask register bits [35:24] and compared against the Offset Base register bits[35:24].

Warning – Changing the Offset Mask or Base register values, while allowed, will cause the location of Fire's address mappings to move in physical address space.

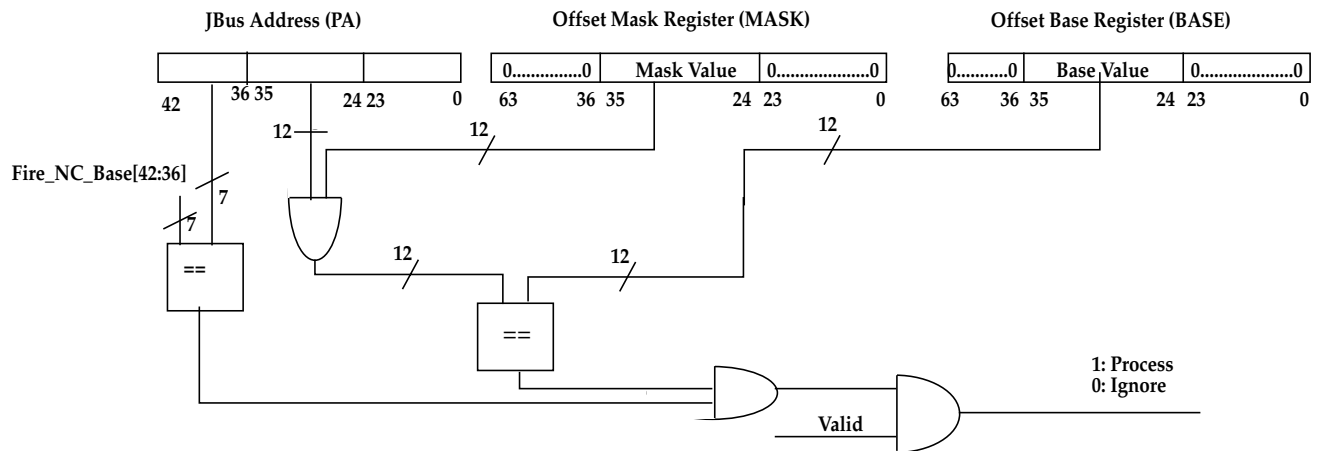
The following sequence may be used to update the Mask and Base registers (assuming that loads are blocking):

- prevent new PIO accesses to mapped space (i.e. quiesce)
- write new values to Mask and Base registers
- perform a read of Mask or Base register (guarantees write has occurred)
- re-enable PIO accesses to mapped space at new location

Note that it is software's job to prevent the PIO accesses from occurring in the first step and to re-enable them (allow them) in the last.

The address filtering is performed as described by the figure below:

Figure 1-2 Address Filtering in Fire



$$\text{Process} = (\text{PA}[42:36] == \text{Fire_NC_Base}[42:36]) \& ((\text{PA}[35:24] \& \text{MASK}) == (\text{BASE})) \& \text{Valid}$$

1.2.2.3 8 MByte Noncacheable Configuration Region

The Configuration Region is further divided into five distinct sections. These subdivisions are summarized in Table 1-7, "8 MByte Noncacheable Configuration Region," on page 28.

Table 1-7 8 MByte Noncacheable Configuration Region

Offset Range	Name	Size	Description
0x00.0000 - 0x3F.FFF8	Fcode	4 MBytes	JBus device ID register (Fire does not support optional Fcode PROM)
0x40.0000 - 0x4F.FFF8	JBus CSRs	1 MByte	JBus CSRs
0x50.0000 - 0x51.FFF8	Reserved Range	0.2 MByte	Not used
0x52.0000 - 0x53.FFF8	I2C CSRs	0.2 MByte	I2C Registers

Table 1-7 8 MByte Noncacheable Configuration Region

Offset Range	Name	Size	Description
0x54.0000 - 0x5F.FFF8	Reserved Range	0.6 MByte	Not used
0x60.0000 - 0x6F.FFF8	PCI Express-A Leaf CSRs	1 MByte	PCI Express-A leaf internal registers, MMU, PCI Express Interrupts, etc.
0x70.0000 - 0x7F.FFF8	PCI Express-B Leaf CSRs	1 MByte	PCI Express-B leaf internal registers, MMU, PCI Express Interrupts, etc.

Note that while the Fcode region exists, Fire does not support an attached PROM for the OBP driver. OBP first reads the COOKIE field of the Device ID register to determine this.

For more information on the registers contained within these subdivisions, see Section 1.3, “CSR Fields and Bits,” on page 1-59.

1.2.2.4 Ebus Sub-region

The Ebus interface provides the ability to do slave PIO cycles on Ebus and provides four separate chip selects. The Ebus sub-region consumes 64 MBytes out of the 64 Gbyte noncacheable region, where each chip select consumes 16 MBytes of this space. See Table 1-8, “Ebus Chip Select Addressing,” on page 29 for more information on EBUS chip select addressing within the Ebus sub-region.

The system boots from an EEPROM attached to chip select 0 on the Ebus. The reset value of the Ebus Offset Base Register bits [35:24] is 0xFF0 and the reset value of the Ebus Offset Mask Register bits [35:24] is 0xFF8 to enable CPU access to the EEPROM through Fire during power-up.

Programmable timing control is provided for EBus chip selects 1 through 3 in the EBus Chip Select x Timing Control registers. The timing control for chip select 0 (Boot PROM chip select) is hardwired until after boot. After boot, programmable timing control is provided in the EBUS EPROM Timing Control Register, once it has been enabled.

The EBus slave interface supports two, four, eight and sixteen byte stacking for EBus devices. The byte stacking is done in little endian format. During byte stacking, a 128 bit, a 64-bit, 32 bit, or 16 bit reads/writes can be initiated from JBus. The EBus controller will break the transfer into multiple byte transactions on the EBus. This mechanism, along with a posted write buffer, allows for better utilization of JBus.

Table 1-8 Ebus Chip Select Addressing

ADDR[26:00]	EBUS Chip Select	Description
0x0XX.XXXX	0	EPROM/Flash PROM
0x1XX.XXXX	1	Generic EBus Device
0x2XX.XXXX	2	Generic EBus Device
0x3XX.XXXX	3	Generic EBus Device

1.2.2.5 I2C Region

Note – The address range, 0x520000-0x5FFFF8, in the 8Mb Noncacheable Configuration Region is dedicated to registers in the I2C cores. Because each core has only 7 addressable registers, there are many addresses in this range that go unused. These addresses are in the sub-ranges 0x520038-0x52FFF8 and 0x530038-0x53FFF8. Be aware however, that if any of these unused addresses are used in transactions, they **will have an effect**. This is because the IRS sub-block receives only bits 16, 5, 4, and 3 of the full address. Based on these bits, the IRS determines where to send transactions. As a result, a transaction sent to a reserved address can cause unintended results. For example, an NCRD sent to the unused address 0x520058 will have the same effect as an NCRD sent to 0x520018, which is a valid address.

1.2.2.6 PCI Express Configuration & I/O Sub-region

There are two PCI Express configuration and IO address sub-regions, one for each PCI Express port. A PCI Express configuration and IO address sub-region is a fixed size of 512 MBytes. The offset within Fire's 64 GByte noncacheable region and sub-region size are set using the PCIE-[A/B] Cfg/IO Offset Base register and the PCIE-[A/B] Cfg/IO Offset Mask register. The offset base and offset mask registers for this sub-region must be initialized for a 512 MByte sub-region. Failure to do so will result in undefined behavior. The first 256 MBytes of the PCI Express configuration and IO address sub-region maps to PCI Express configuration space. The last 256 MBytes of this sub-region maps to PCI Express IO space. All configuration and IO packets are issued with the virtual channel ID equal to zero (the default virtual channel).

The mapping from the s/w relative (i.e. byte addressing, no byte masks) physical address [42:00] to the PCI Express Configuration & I/O sub-region is as follows:

- 42:41 - set to 0x3 (2'b11) to indicate a 64 GByte noncacheable region access
- 40:36 - matches one of Fire's JBus AgentIDs (Fire uses two AgentIDs)
- 35:29 - Configuration & I/O Address Match. These bits determine if the access to Fire maps into one of the Configuration & I/O sub-regions.
- 28 - Configuration or I/O operation. 0x0 (1'b0) signifies a configuration operation. 0x1 (1'b1) signifies a I/O operation.
- 27:00 - Configuration or I/O space specific addressing information. See Section 1.2.2.6.1, "Accessing PCI Express Configuration Space and Section 1.2.2.6.2, "Accessing PCI Express I/O Space for more information on how these address bits are used.

Note – All valid noncacheable JBus reads & writes which are 4 bytes or less are accepted by Fire when accessing the PCI Express Configuration & I/O sub-region. An ILL_ACC error will be reported in the JBC Error Status Clear Register for invalid write accesses to the PCI Express Configuration & I/O sub-region.

1.2.2.6.1 Accessing PCI Express Configuration Space

The mapping from the 28-bit physical address offset within the PCI Express Configuration space to the Configuration Addressing information is as follows:

- 27:20 (8) - PCI Express Bus Number
- 19:15 (5) - Device Number

- 14:12 (3) - Function Number
- 11:00 (12) - PCI Express Configuration Register Address. Address bits [01:00] are used to determine the byte enables on PCI Express.

Note – The Configuration Address Space per Device Function has been extended in PCI Express from 256 bytes to 4096 bytes. Configuration accesses through Fire are different from previous generation Sun Bridges. The Configuration Register Address has been extended to 12 bits (from 8) and the bus number, device number, and function number have been shifted left 4 bits. Therefore, software which accesses configuration registers for PCI Express devices and PCI devices sitting behind a PCI Express to PCI bridge, will be different on Fire based systems.

If the Bus Number for a configuration access matches the Bus Number in "DMC PCI Express Configuration" Register, a Type 0 Configuration Packet is generated. The Bus Number in "DMC PCI Express Configuration" Register is programmable by software. If the Bus Number does not equal to the Bus Number in DMC PCI Express Configuration Register, a Type 1 Configuration Packet is generated.

A PCI Special Cycle can be generated on a PCI bus attached (directly or indirectly) to a PCI Express to PCI bridge by sending a Type 1 Configuration Packet with the appropriate bus number, all ones for the Device Number (0x1F) and Function Number (0x7), and all zeros for the Configuration Address; e.g. PartialAddress[19:00] = 0xFF000.

Fire does not implement any registers within PCI Express configuration space.

See the PCI and PCI Express specifications for more detail on accessing Configuration Space.

1.2.2.6.2 Accessing PCI Express I/O Space

The mapping from the 28-bit physical address offset within the PCI Express I/O space to the PCI Express I/O address is as follows:

- 27:0 - PCI Express I/O Address. Address bits 31:28 are always set to zero.

1.2.2.7 PCI Express 32-bit Addressing Memory Sub-region

There are two PCI Express 32-bit Addressing Memory sub-regions, one for each PCI Express port. A PCI Express 32-bit Addressing Memory sub-region is a programmable size ranging from 16 MBytes to 2 GBytes. The offset within Fire's 64 GByte noncacheable region and sub-region size are set using the PCI Express Bus[A/B] Mem32 Offset Base register and the PCI Express Bus[A/B] Mem32 Offset Mask register. The offset base and offset mask registers for this sub-region must be initialized for a sub-region between 16 MBytes and 2 GBytes. Failure to do so will result in undefined behavior.

The mapping from the s/w relative (i.e. byte addressing, no byte masks) physical address [42:00] to the PCI Express 32-bit Addressing Memory sub-region is as follows:

- 42:41 - set to 0x3 (2'b11) to indicate a 64 GByte noncacheable region access
- 40:36 - matches one of Fire's JBus AgentIDs (Fire uses two AgentIDs)

- 35:(X+1) - Address Match. These bits determine if the access to Fire maps into one of the 32-bit Addressing Memory sub-regions. X is determined by the size of the Mem32 sub-region which is set by the PCI Express Bus[A/B] Mem32 Offset Mask register. See Section 1.2.2.2, “Offset Base and Offset Mask Register Behavior,” on page 1-27 for more information.
- X:00 - PCI Express 32-bit memory address. The PCI Express Mem32 address bits 31:(X+1) are set to zero. X is determined by the size of the Mem32 sub-region which is set by the PCI Express Bus[A/B] Mem32 Offset Mask register. See Section 1.2.2.2, “Offset Base and Offset Mask Register Behavior,” on page 1-27 for more information.

Note – All valid noncacheable reads & writes and noncacheable block reads & block writes on JBus are accepted by Fire when accessing the PCI Express 32-bit Addressing Memory sub-region.

1.2.2.8 PCI Express 64-bit Addressing Memory Sub-region

There are two PCI Express 64-bit Addressing Memory sub-regions, one for each PCI Express port. A PCI Express 64-bit Addressing Memory sub-region is a programmable size ranging from 16 MBytes to 32 GBytes. The offset within Fire’s 64 GByte noncacheable region and sub-region size are set using the PCI Express Bus[A/B] Mem64 Offset Base register and the PCI Express Bus[A/B] Mem64 Offset Mask register. The offset base and offset mask registers for this sub-region must be initialized for a sub-region between 16 MBytes and 32 GBytes. Failure to do so will result in undefined behavior. All downbound packets to this sub-region are issued with the virtual channel ID equal to zero (the default virtual channel).

The mapping from the s/w relative (i.e. byte addressing, no byte masks) physical address [42:00] to the PCI Express 64-bit Addressing Memory sub-region is as follows:

- 42:41 - set to 0x3 (2'b11) to indicate a 64 GByte noncacheable region access
- 40:36 - matches one of Fire’s JBus AgentIDs (Fire uses two AgentIDs)
- 35:(X+1) - Address Match. These bits determine if the access to Fire maps into one of the 64-bit Addressing Memory sub-regions. X is determined by the size of the Mem64 sub-region which is set by the PCI Express Bus[A/B] Mem64 Offset Mask register. See Section 1.2.2.2, “Offset Base and Offset Mask Register Behavior,” on page 1-27 for more information.
- X:00 - Offset within PCI Express 64-bit memory address.

The PCI Express Mem64 PIO address is formed from

{offset[63:36], (offset[35:24] | jbus addr[35:24]), jbus addr[23:2]}

where jbus addr[35:0] = {(PA[35:24] & ~mask[35:24]), PA[23:0]} and offset[63:24] is from the Mem64 PCIE Offset Register.

Note – All valid noncacheable reads & writes and noncacheable block reads & block writes on JBus are accepted by Fire when accessing the PCI Express 64-bit Addressing Memory sub-region.

1.2.2.9 JBus Interrupt Space

Fire ignores and silently discards any interrupts sent to it (which should never happen). The system could potentially hang if an interrupt is sent to Fire.

1.2.3 Fire to JBus (Upbound) Transactions

1.2.3.1 PCI Express => Fire

PCI Express defines the following four address spaces:

- PCI Express Configuration Space
- PCI Express I/O Space
- PCI Express Memory Space
- PCI Express Message Space

Fire does **not** respond to PCI Express Configuration space or I/O space transactions.

Fire does respond to PCI Express memory space transactions. This is the space in which DVMA and DMA (MMU bypass) activity takes place. Fire does not support peer-to-peer transactions, however PCI Express switches or PCI Express-to-PCI bridges may support peer-to-peer transactions which would require no interaction with Fire. See “PCI Express Memory Transaction Processing” on page 1-33 for more information on how Fire handles PCI Express memory space transactions.

PCI Express defines one set of message groups which live within the PCI Express message space; Standard Messages. Fire will generate and/or respond to a subset of the Standard Messages. See “PCI Express Message Transaction Processing” on page 1-34 for more information on how Fire handles Standard Messages.

1.2.3.1.1 PCI Express Memory Transaction Processing

The final destination and address translation of a PCI Express Memory transaction is based on:

- PCI Express addressing mode used: 64-bit vs. 32-bit
- PCI Express address bit <31> when a 32-bit address is used
- Value of Translation Enable (TE) in the MMU Control Register
- Value of Bypass Enable (BE) in the MMU Control Register
- Value of PCI Express address bits <63:42> when a 64-bit address is used

Table 1-9, “PCI Express Memory Transaction Behavior,” on page 33 shows the various ways that Fire, as a PCI Express target device, deals with PCI Express Memory Space addresses.

Table 1-9 PCI Express Memory Transaction Behavior (Sheet 1 of 2)

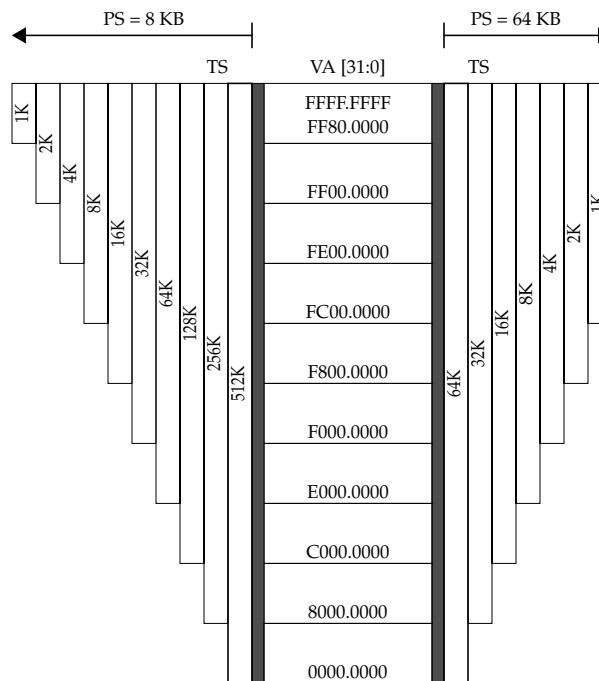
Mode	TE	BE	Addr[63:42]	Addr[31:0]	Result
32-bit	0	X	N/A	N/A	MMU TRN_ERR
32-bit	1	X	N/A	Figure 1-3	Figure 1-3
64-bit	X	0	X	N/A	MMU BYP_ERR

Table 1-9 PCI Express Memory Transaction Behavior (Sheet 2 of 2)

Mode	TE	BE	Addr[63:42]	Addr[31:0]	Result
64-bit	X	1	22'b0000_0000_0000_0000_0000_00 - 22'b1111_1111_1111_1011_1111_11	N/A	MMU BYP_OOR
64-bit	X	1	22'b1111_1111_1111_1100_0000_00	N/A	Bypass (DMA)
64-bit	X	1	22'b1111_1111_1111_1100_0000_01 - 22'b1111_1111_1111_1111_1111_11		MMU BYP_OOR

The size of the TSB table determines the range on the virtual addresses which are valid. Addresses not in the valid range cause a translation out-of-range error (TRN_OOR). TSB sizes and page sizes which are not diagrammed in this figure are not valid and produce undefined results.

Figure 1-3 Translation Size Virtual Address Configurations



In MMU translation mode, the physical address is obtained by performing a virtual to physical translation through the MMU. See “MMU and Bypass Operation” on page 1-36 for more information on MMU operation.

Fire does not support peer-to-peer operations between I/O devices on JBus. PCI Express switches and PCI Express-to-PCI bridges may support peer-to-peer operations.

In bypass mode, JBus<42:0> = PCI Express<42:0>. (<42:0> is a logical representation of the address bits).

1.2.3.1.2 PCI Express Message Transaction Processing

PCI Express defines the following message groups within the Standard Messaging group:

- Interrupt Signaling
- Locked Transaction Support

- Error Signaling
- Power Management
- Slot Power Limit Support
- Payload Defined
- Vendor Specific
- Hot Plug Signalling

The Interrupt Signalling message group consists of the Assert_INTx message and the Deassert_INTx message. MSIs are implemented via a memory write transaction. They are not messages and therefore, not included with the Interrupt Signalling messages. Fire supports both Interrupt Signalling messages and MSIs. See Section 1.2.6.8.2, “Message Signaled Interrupts (MSIs and MSI-Xs),” on page 1-49 for more information about MSIs.

Locked Transaction messages are **not** supported by Fire

The Error Signalling message group consists of Correctable Error detected (ERR_COR), Uncorrectable Error detected (ERR_UNC), and Fatal Error Occurred (ERR_FATAL). Fire supports all error messages. See Section 1.2.6.8.1, “PCI Express Messages,” on page 1-49 for more information on Error Signalling message handling.

The Power Management message group consists of the following messages:

- PM_Active_State_NAK
- PM_PME
- PME_Turn_Off
- PME_TO_Ack

All Power Management messages are supported by Fire. See “PCI Power Management” on page 1-51 for more information on Power Management message handling. Fire does not support powering on a system from a PME message.

All Ordering Control Primitive messages which are required to make Fire sun4u compliant are supported by Fire. Software has no visibility into the occurrence of these messages, therefore they will not be discussed further in this document.

Any write to TLU Slot Capabilities Register will cause Fire to send a set_slot_power_limit message when link is up. It will use the Slot Power Limit Scale and Slot Power Limit Value fields in the TLU Slot Capabilities Register when generating the message. These fields are programmable by software.

Fire does not support Payload Defined messages (messages with data) or vendor specific messages. Fire will silently drop Vendor-Defined Type 1 messages as required by the PCI Express specification.

The Hotplug message group consists of the following messages:

- Attention_Indicator_On
- Attention_Indicator_Blink
- Attention_Indicator_Off
- Power_Indicator_On
- Power_Indicator_Blink
- Power_Indicator_Off
- Attention_Button_Pressed

Fire does not support the Hotplug message group. If a system requires these messages, they can use a switch which supports them. See section 2.8.1.8. Hot Plug Signaling Messages on page 96 of the PCI Express Specification v1.0(RC1) for more information on switch support for PCI Express Hotplug messages.

All messages which are not supported will result in the logging of an Unsupported Request.

1.2.4 MMU and Bypass Operation

The MMU provides a virtual to physical address translation on a range of upbound PCI Express transactions. See Table 1-9, “PCI Express Memory Transaction Behavior,” on page 33 for more information on which transactions are translated using the MMU. The MMU allows a collection of discontinuous physical memory pages to be represented as a contiguous virtual I/O address range. It also provides a level of protection by detecting a subset of invalid memory reads and writes to addresses which are marked as invalid IO addresses.

1.2.4.1 Translation Storage Buffer Overview

The MMU fetches translation information from a Translation Storage Buffer (TSB) in system memory. The TSB consists of an array of 8 byte Translation Table Entries (TTE) which provide mapping information for the virtual DMA pages.

The base size of a page that a single TTE maps to is set while initializing the MMU. This can be set to either an 8KByte or 64 KByte page. The base page size is used by the MMU to determine the index into the TSB (i.e. bits 15:00 of the address are ignored when calculating the index into the TSB for a base page size of 64 KBytes, bits 12:00 of the address are ignored for an 8 KByte page).

The size of the TSB can be initialized from 1K to 512K entries in power of 2 increments. Therefore, the TSB must reside in a physically contiguous buffer ranging from 8KBytes to 4MBytes since each entry consumes 8 bytes. The DVMA address space available for an 8 KByte base page, ranges from 8 MBytes to 4 GBytes depending on the size of the TSB. The DVMA address space available for a 64 KByte base page, ranges from 64 MBytes to 4 GBytes depending on the size of the TSB (a DVMA address space larger than 4 GBytes is not supported by Fire).

Fire does **not** support mixed page size TTEs (i.e. both 8 KByte and 64 KByte TTEs in the TSB at the same time). This feature is no longer required since the TSB cache is cacheline based now (previous bridges cached individual TTEs). Existing code will still function correctly with no performance degradation (i.e. this change is backwards compatible). See Section 1.2.5.1, “TSB Cache Operation, Initialization, and Debug,” on page 1-39 for more information on the TSB cache.

The base address of the TSB table has to be aligned on 8K boundary. The lower order 13 bits are assumed to be 0x0. Tables larger than 8K bytes are only constrained to be on 8K boundaries rather than having to be size aligned.

1.2.4.1.1 TSB Indexing

An index into the TSB is calculated from the DVMA address (VirtAddr) as follows:

```

if (base page size == 8KB)
    Page Bits = 13;
else
    Page Bits = 16;
Index = (VirtAddr >> Page Bits) & (Table Size - 1);

```

Table 1-10 TSB Indexing

TSB Size	8 KByte Base Page		64 KByte Base Page	
	VA Space Size	TSB Index [3]	VA Space Size	TSB Index [3]
1K Entries	8 MB	VA<22:13>	64 MB	VA<25:16>
2K Entries	16 MB	VA<23:13>	128 MB	VA<26:16>
4K Entries	32 MB	VA<24:13>	256 MB	VA<27:16>
8K Entries	64 MB	VA<25:13>	512 MB	VA<28:16>
16K Entries	128 MB	VA<26:13>	1 GB	VA<29:16>
32K Entries	256 MB	VA<27:13>	2 GB	VA<30:16>
64K Entries	512 MB	VA<28:13>	4 GB	VA<31:16>
128K Entries	1 GB	VA<29:13>	not allowed ¹	--
256K Entries	2 GB	VA<30:13>	not allowed ¹	--
512K Entries	4 GB	VA<31:13>	not allowed ¹	

1. Hardware does not prevent illegal combinations from being programmed. Programming an illegal combination into the MMU will result in undefined behavior.

1.2.4.2 Translation Table Entry (TTE)

A TTE entry has valid information only when the Valid bit (DATA_V) is set. TTEs have the following format:

Table 1-11 TTE Data Format (Sheet 1 of 2)

Bits	Field	Description
63	DATA_V	Valid bit (1 = TTE entry has valid mapping)
62	Reserved	Reserved for software use, is ignored by hardware.
61	Reserved	Reserved for software use, is ignored by hardware.
60	Reserved	Reserved for software use, is ignored by hardware.
59:47	Reserved	Reserved for software use, is ignored by hardware.
46:43	Reserved	Reserved for software use, is ignored by hardware.
42:13	DATA_PA	Contains bits [42:13] of physical address. Bits 15:13 are not used for 64K page. Bits [42:41] should be initialized to 2'b00 to indicate a cacheable JBus transaction. Failure to do so will result in undefined behavior.
12:7	DATA_SOFT	Reserved for software use, is ignored by hardware.
6:5	Reserved	Reserved for software use, is ignored by hardware.

Table 1-11 TTE Data Format (Sheet 2 of 2)

4	Reserved	Reserved. Should be ignored by hardware for software backward compatibility.
3:2	Reserved	Reserved for software use, is ignored by hardware.
1	DATA_W	Set if this page is writable
0	Reserved	Reserved for software use, is ignored by hardware.

Note – The DATA_SIZE, LOCALBUS, CONTEXT, and CACHEABLE fields have been removed from the TTE for Fire. They are marked as "Reserved. Should be ignored by hardware for software backward compatibility" in order to maintain compatibility with any software which may use these fields. DATA_SIZE is no longer required since the H/W will cache a cacheline of TTEs (i.e. there is no benefit to having 64K pages in 8K TTE entries). LOCALBUS is not meaningful for PCI Express. CONTEXT was used to manage a software coherent TSB cache, which is not present in Fire. Fire does not support generating non-coherent JBus transactions, therefore the CACHEABLE field is not required.

1.2.5 Bypass and MMU Initialization

There are two paths that an upbound transaction can take based on the memory transaction addressing mode. Fire will attempt to translate (using the MMU) a 32-bit mode memory transaction and "bypass" a 64-bit mode memory transaction. Each of these paths can be individually enabled/disabled without side effects on the other. See Table 1-9, "PCI Express Memory Transaction Behavior," on page 33 for more information how upbound memory transactions are handled.

Bypass is disabled/enabled via the Bypass Enable (BE) bit in the MMU Control Register. When Bypass is enabled, upbound 64-bit mode memory transactions with the upper 14 address bits [63:50] set to all 1's, address bits [49:43] set to zero, and address bit [42] set to zero (Fire only supports generating cacheable JBus transactions), are converted to the appropriate JBus operation. The lower 42 bits of the address [41:0] are used as the cacheable physical address on JBus.

If bypass is disabled, or if address bits [63:42] are not set to the correct values stated earlier, an error is flagged which optionally generates an interrupt (A 64-bit address which is not addressed to Fire should never reach Fire since PCI Express links are point to point, e.g. a switch should never forward a packet to Fire which is meant for another device).

Note – The Bypass control bit has been changed from previous generation Sun bridges. The Bypass control bit has been moved from a register not in the MMU to the MMU control register and been changed to an enable bit (from a disable bit). The default (after cold and warm reset) setting of Bypass has been changed to disabled. Previous generation bridges do not verify that address bits 49:43 are set to zero and have the capability to generate non-cacheable JBus transactions for cross bus peer to peer operations. Non-cacheable JBus transactions are not generated in practice because they lead to deadlock conditions, hence the removal from Fire.

The MMU is disabled/enabled via the Translation Enable (TE) bit in the MMU Control Register. When the MMU is enabled, upbound 32-bit mode memory transactions will attempt to be translated to the 43 bit [42:0] cacheable physical address on JBus. To translate, an index into the TSB is calculated (see “TSB Indexing” on page 1-36 for more information) from the IO address, and the TTE is fetched. If the TTE is valid, Fire will check to see if the transaction is a write and the page is marked as writable in the TTE. If the transaction is a read, or is a valid write transaction, Fire will convert the transaction to the appropriate JBus operation using the physical address specified in the TTE.

If the MMU is disabled, or if one or more of the assertions stated above are not valid, an error is flagged which optionally generates an interrupt. If the TTE is invalid, or the TSB does not cover the address in question (i.e. the TSB size is less than maximum size and the address is outside the valid address range specified by TSB_SIZE), an error is generated.

Note – The MMU Enable behavior has changed from previous generation Sun bridges. If the MMU is disabled, it will no longer pass through DMA traffic. Instead it will ignore upbound reads and writes, log an error, and optionally generate an interrupt.

During normal operation, bypass and the MMU should be initialized as follows:

- Translations should be enabled (1)
- Bypass should be enabled (1)
- Process Disable should be enabled (0)

1.2.5.1 TSB Cache Operation, Initialization, and Debug

The TSB resides in system memory and contains TTEs which provide mapping information for the virtual DMA pages. See “Translation Storage Buffer Overview” on page 1-36 for more information on the TSB.

Fire implements a hardware coherent cache to locally store recently accessed TTEs in order to reduce DVMA transfer latencies. Fire never modifies the contents of the TTEs therefore, will never write back the contents of the cache to system memory. Cache eviction is based on a pseudo-LRU algorithm. Fire supports a software flush mechanism to support a non coherent JBus implementation via the PCIE-A MMU TTE Cache Flush Address Register and the PCIE-B MMU TTE Cache Flush Address Register.

Fire caches reads to the TSB. A TSB cache entry is based on the JBus cacheline size, which is 64 bytes. Each cacheline can hold up to 8 valid TTE entries. Up to 64 cache entries can be stored in the TSB cache at any given time giving a maximum total of 512 TTE's which can be cached.

Fire will invalidate a TSB cache entry on a WRIS, WRI, INV, OWN, or a RDO transaction. Fire will return shared for all Read-To-Share (or other JBus operations which require a shared response) accesses to the TSB, even if the cacheline is not present within Fire's TSB cache (due to a performance optimization within Fire). Therefore, a CPU will not get exclusive access to a cacheline within the TSB when it does a Read-To-Share.

Software may need to use a more intelligent TTE allocation algorithm to realize the full performance potential of the hardware coherent TSB cache. Since each entry in the TSB cache consists of 8 TTEs, initializing one TTE entry could invalidate 7 other TTE entries currently in the cache (i.e. they would be removed from the cache, but still would be "valid" TTEs). Therefore, a simple sequential allocation scheme may not be the best approach. It is important to note that the TTE allocation algorithm will have to balance efficient cache utilization with fragmentation issues that arise when you sparsely populate the cachelines.

There are three bits which control the behavior of Fire's TSB cache. They are TSB Cache Mode, CM, (2 bits) and TSB Snoop Enable, SE, (1 bit). When Translation is disabled, TSB Cache Mode has no meaning and is ignored. TSB Cache Snoop Enable is independent of Translation Enable and TSB Cache Mode behavior.

During normal operation, the TSB cache control bits should be initialize to the following:

- TSB Cache Mode should be enabled (3)
- TSB Cache Snoop should be enabled (1)

The TSB cache can only be "read from"/"written to" via PIOs when the TSB cache is disabled. The Translation enable setting has no effect on accesses to the TSB cache. Attempts to read from or write to the TSB cache when it is enabled are flagged as an error and optionally generates an interrupt.

Note – The TTEs can be initialized to valid settings for debug purposes. See the TSB Cache Mode and TSB Snoop Enable write-up below for more information on this behavior.

Table 1-12 MMU TSB Cache Mode

CM	Description
00	TSB cache is disabled. A single cacheline, hardware coherent TLB is used to cache the last used TTE cacheline.
01	TSB cache is enabled and locked. Misses cause an error. The TSB in main memory is not used.
10	TSB cache is enabled and locked. Misses use the TLB.
11	TSB cache is enabled (default)

If Translation is enabled, and the TSB cache is disabled (CM=0) Fire uses a single cacheline, hardware coherent TLB to cache the last used TTE. If TSB Snoop Enable is disabled, software must flush the TLB entry to ensure that the TLB is consistent with the TSB during unmap operations. There are no restrictions on when the TLB entry can be flushed. It is expected that this functionality will only be used for bringup/debug if issues with the hardware coherent TSB cache are discovered.

If TSB Snoop Enable is disabled, Fire will not invalidate (or returned shared for read-to-share transactions) the TSB cache entries based on JBus accesses to the TSB. TSB Snoop Enable has no effect on hits on the TSB cache entries or misses, evictions, and loads into the TSB cache.

1.2.6 Interrupt Model

PCI Express supports two different interrupt mechanisms; INTx emulation and Message Signaled Interrupts (MSI). INTx refers to four PCI Interrupt pins, named INTA-D, abbreviated as INTx. On PCI, each device can assert one interrupt per function, with a total of four discreet interrupt pins or wires. The wiring of these interrupt signals on PCI is platform and system specific. MSIs are defined in PCI 2.2 as an optional feature, but have not been widely adopted by PCI devices since a PCI device must also support INTx in order to work in all existing systems. MSI support is required in PCI-X devices.

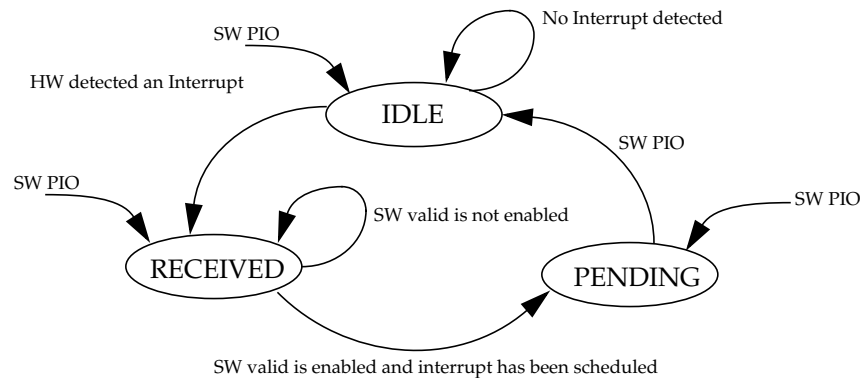
INTx emulation provides software compatibility for firmware and bridge drivers (assuming the PCI Express bridge is register compatible with an existing PCI bridge). However, INTx emulation is expected to be deprecated in future versions of the PCI Express specification. Message Signaled Interrupts (MSI) are mandatory in PCI Express and are software compatible with existing PCI device drivers.

MSIs are inband posted (no acknowledge generated) writes to a pre-programmed address (either 32-bit or 64-bit). MSIs can be generated from PCI Express devices or PCI-X devices behind a PCI Express to PCI-X bridge. Interrupts on the various Sun processor buses are also inband "writes". However, a processor can NAK an interrupt causing the interrupter to have to resend the interrupt at a later time.

When Fire generates an interrupt, the source of the interrupt could be from one of Fire's internal blocks (e.g. i2c or an error occurred); from an external device hooked up to the internal interrupt concentrator; from a PCI Express INTx Assertion Packet; or from one of Fire's Event Queues (due to one or more MSIs, Power Management messages, error messages, hotplug messages, etc. being received). All interrupt sources are maskable.

Fire uses a subset of interrupt state machine in Schizo and Tomatillo. An interrupt is always a "level" sensitive interrupt. An interrupt can be in one of three states: IDLE, RECEIVED, or PENDING. IDLE represents the state where no interrupts are reported. RECEIVED indicates that an interrupt has been detected and should be delivered to the processor if/when the valid bit is set in its mapping register. PENDING is the state when the interrupt has been queued to be or has been sent to the processor. Any subsequent detection of the same interrupt is ignored until software resets the state machine back to IDLE. The state register for each level sensitive interrupt can be set to any desired state by software via the Interrupt Clear Registers.

Note – Software should ensure that the source of a level sensitive interrupt is cleared before clearing the interrupt state register via the Interrupt Clear Register, otherwise Fire will continue to reissue the interrupt each time the state register is set to IDLE.



1.2.6.1 Internal Interrupts

Each PCIE leaf can generate four internal interrupts, 2 I2C interrupts (INO 60 and 61), a PCIE internal interrupt (INO 62), and a JBus block internal interrupt (INO 63). The JBus-block internal interrupt and the I2C interrupts are routed to both PCIE leaves. Software should only enable the I2C and JBus interrupts in a single PCIE leaf. Failure to do so will cause unexpected behavior. The PCIE internal interrupt can be enabled in both leaves.

1.2.6.2 External Out of Band Interrupts

Fire supports 40 external (out of band) interrupts from external pins routed into an internal interrupt concentrator in the JBC core. Each external interrupt has a dedicated JBus interrupt mondo. Out of the 40 interrupts, interrupt pins 0-19 are dedicated to the PCIE-A leaf [INOs 0-19], and interrupt pins 20-39 are dedicated to the PCIE-B leaf [INOs 0-19].

1.2.6.3 PCI Express INTx Emulation

Fire generates four level sensitive signals based on state bits which are asserted when Fire sees the appropriate Assert_INTx message and deasserted when it sees the appropriate Deassert_INTx message. Each of the four INTx interrupts have a dedicate JBus interrupt mondo.

From the PCI Express Base Specification, Rev. 1.0a [04/15/03], 2.2.8.1. INTx Interrupt Signaling - Rules, pg 65.

"The Assert_INTx/Deassert_INTx Message pairs constitute four "virtual wires" for each of the legacy PCI interrupts designated A, B, C, and D. The following rules describe the operation of these virtual wires:

- The components at both ends of each Link must track the logical state of the four virtual wires using the Assert/Deassert Messages to represent the active and inactive transitions (respectively) of each corresponding virtual wire.
- An Assert_INTx represents the active going transition of the INTx (x = A, B, C, or D) virtual wire.

- A Deassert_INTx represents the inactive going transition of the INTx (x = A, B, C, or D) virtual wire.

Note – Duplicate Assert_INTx/Deassert_INTx Messages have no effect, but are not errors.

Note – If a PCI-E link goes down, software must clear the INTx state (related to that port) to ensure consistent INTx state.

INTx legacy interrupts are inband, so they can be stalled due to data traffic. An interrupt deassert could be late because it was stalled, it may not be coming because the device did not deassert the interrupt, it may not be coming because the interrupt is shared by another device. For the second two cases, S/W will be interrupted again.

INTx legacy interrupts should be treated just like the external interrupts with the additional caveat that you may get spurious INTx legacy interrupts due to a late arriving INTx deassert message.

It is **STRONGLY** encouraging that any devices with discrete interrupts (e.g. systems w/ PCI devices or slots behind a PCI-E to PCI-X bridge), be wired up to Fire's interrupt concentrator or use MSIs.

1.2.6.4 *Event Queue Interrupts*

Fire uses event queues to queue up MSIs and valid PCI Express messages received which require software notification. An event queue is tied to a specified processor and generates only one outstanding JBUS interrupt mondo for one or more writes to the event queue. Multiple event queues exist primarily to provide hardware based interrupt distribution among the processors. However, it is possible for software to use an additional event queue for just error messages, for example.

Internal to Fire, each event queue exports a level signal (called NOT_EMPTY for this discussion) which is asserted whenever head != tail. NOT_EMPTY is deasserted if head == tail.

A processor interrupt mondo will be sent to the software programmable target processor if the event queue is enabled, NOT_EMPTY is asserted, and the event queue's interrupt state is IDLE. Each Event Queue has a dedicated JBus interrupt mondo.

Event queues are aligned on a 64-byte boundary. An entry in the event queue is called an event queue record. Event queue records are 64-byte aligned and are 64-bytes large. The base address of an event queue is an MMU address. It can be a 32-bit virtual IO address or a 64-bit physical address (using MMU bypass). If it is a bypass address, the queue must reside in physically contiguous memory. The head pointer and tail pointer are index's into the event queue.

Items in the queue are written to the end of the queue using the tail index. Items are removed from the front of the queue using the head index (An item enters the queue at the end of the queue and is removed from the queue when it reaches the front of a queue). "With queues, we speak of the front and the rear of the queue; things enter at the rear and are removed when they ultimately reach the front position." - Knuth

When the head index and the tail index are equal, the queue is considered empty. When $(tail + 1) == head$, the queue is considered full. This means the EQ is full at 127 entries. If Fire attempts to write to the event queue when it is full, it will drop the write, set the overflow bit, and generate an internal interrupt signaling the error.

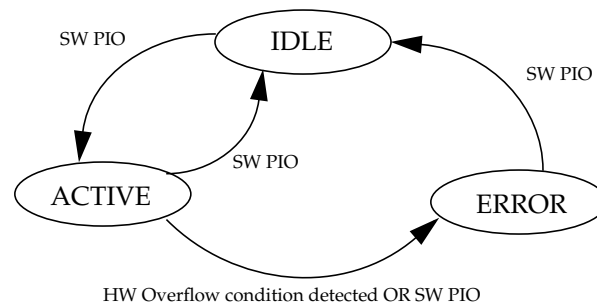
Fire writes event queue records to the queue using the event queue address indexed by the value in tail. In Fire, the tail is incremented BEFORE the write has successfully completed. Therefore, the event queue's tail register cannot be used by software to determine the number of event queue records present in the event queue. Fire never updates the value in head. The event queue head is only writable by software.

System software reads an event record by reading the event queue record at the address of the queue indexed by the value in head. The event queue head may only be incremented after the system software successfully reads one or more event records. Once initialized by system software, the value in tail is never updated by system software. The event queue head is updated by writing a new value to the event queue's head register. Since the actual head is written to the event queue's head register, more than one event queue record can be "freed up" with a single write to the register.

Since software cannot use the tail register to determine the number of event queue records present in the event queue, it is suggested that software uses the first 64-bit value in the event queue record to determine when it has processed the last event queue record. To accomplish this, software should zero the event queue before use and then look for a nonzero value in the first 64-bit value in the event queue record to indicate if an event queue record is present at the current head. If the head contents are zero, the event queue is empty. Therefore, when software processes an event queue record, it must zero out the first 64-bit value in the event queue BEFORE writing to the event queue head register to indicate it has processed the event queue record.

For Fire, all event queues live in a contiguous memory region, where the first event queue must be aligned on a programmable 512 Kbyte boundary. Each event queue has a fixed size of 128 entries (one 8 Kbyte page) but is considered full at 127 entries.

Each event queue has the following state included; enable/disable event queue, head index, tail index, overflow error occurred, and the event queue state (IDLE, ACTIVE, ERROR). Software can induce any of state transitions with the exception of IDLE to ERROR.



1.2.6.5 JBUS Interrupt Mondos

A JBUS interrupt mondo consists of an address and eight 64-bit data words. The address identifies the JBUS packet as an interrupt packet and also identifies the target CPU which will receive the interrupt.

Each PCI Express leaf can generate 64 unique interrupt mondo's on JBUS. The 64 interrupt mondos break down as follows. There are 20 interrupts from the internal interrupt concentrator (INO 0-19), 4 interrupts for PCI Express INTx Emulation (INO 20-23), 36 event queue interrupts (INO 24-59), two I2C interrupts (INO 60 & INO 61), a PCIE leaf internal interrupt (INO 62), and an internal JBus block interrupt (INO 63). Each interrupt uses the interrupt state machine specified in Section 1.2.6, "Interrupt Model," on page 1-41.

Fire can send two differently formatted interrupt mondo's based on the setting of MDO_MODE in the Interrupt Mapping registers. The interrupt mondo format is specified below (The last six 64-bit data words are always set to 0).

```

if (MDO_MODE == 0)
    MDATA0[63:0] = {53'h0, fire_jpid[4:0], ino[5:0]}
    MDATA1[63:0] = {64'h0}
} else {
    MDATA0[63:0] = {data0[63:6], ino[5:0]};
    MDATA1[63:0] = {data1[63:0]}
}
  
```

*1 - fire_jpid[4:0] => this ID is unique between the two PCI-E leafs on Fire.

*2 - data0 => Interrupt Mondo Data 0 Register (0x0062C000, 0x0072C000 / 0x0)

*3 - data1 => Interrupt Mondo Data 1 Register (0x0062C008, 0x0072C008 / 0x0)

Note – To properly use event queues, the sun4u interrupt trap handler should support, directly or via a device specific extension, reading event queue records and scheduling interrupt handlers (based on the event queue records) to run at the appropriate interrupt level. An event queue can have multiple event queue records present at any given time. Each event queue record corresponds to a unique interrupt with its own interrupt level (i.e. an event queue can have event queue records enqueued which will run at different interrupt levels).

Fire also supports 4 interrupt groups which allow multiple interrupts to be issued at a time. Schizo & Tomatillo can only issue one interrupt at a time. JBus interrupt mondos are assigned to an interrupt group. A given interrupt group can only have one interrupt outstanding at a time (i.e. until a processor acknowledges the interrupt). It is expected that s/w will use one interrupt group per processor present in the system.

Note – When using the group controller functionality in Fire, it is very important that the group controllers and the interrupt mapping registers be set up properly. Failure to do this can cause dropped interrupts, Mondo time-out errors, unsolicited ACK/NACK errors and possible interrupt dead lock. Fire posts outstanding mondos to its JBUS scoreboard using the (Target ID, Source ID) for the Mondo as its key into the CAM lookup. Since the Source ID will not change for Fire and is hard coded based on Fire's Master/Slave configuration and which Leaf the mondo is generated by, if two mondos were to be generated to the same Target ID at the same time the JBUS scoreboard would be come corrupted which could then cause the error conditions listed previously. Therefore in order to avoid these problems the following rule should be followed: ALL MONDOS WITH THE SAME TARGET ID PROGRAMMED IN THE INTERRUPT MAPPING REGISTERS MUST ALSO BE PROGRAMMED TO USE THE SAME GROUP CONTROLLER IN THE INTERRUPT MAPPING REGISTERS.

JBUS Interrupt Mondo State includes; valid/invalid entry, current interrupt state, Target Processor Agent ID, bit 63 for DATA0 (1-bits), and DATA1 (64-bits), interrupt group.

1.2.6.6 *Interrupt Relocation*

Not all of the event queues have to be used. In normal system operation, it is expected that there will be no more than one event queue active per processor in the system (at least for handling MSIs). Software typically retargets interrupts when more CPUs come online or when some CPUs go offline. Both conventional discreet interrupts and MSIs may be retargeted to different CPUs upon software request.

To retarget an interrupt, software must clear the valid bit of the interrupt mapping register, and then busy wait until the interrupt state is no longer PENDING. If an interrupt is not in progress when the valid bit is cleared, this is immediately satisfied. If an interrupt is currently being handled, software may wait quite a while. Once the interrupt state is IDLE, software can then change the CPU ID field of the mapping register to a new CPU ID, and then renable the interrupt by setting the valid bit in the interrupt mapping register.

A MSI cannot be dynamically relocated to a new event queue unless software can ensure that the MSI is not in progress and that the device will not send that particular MSI while the MSI is the process of being retargeted.

1.2.6.7 Data flushing

No special sync or sync flush operations within Fire are required to synchronize data with interrupts. Fire flushes all DMA writes in PCI Express's virtual channel #0 before returning the response to a PIO read. PCI to PCI Express bridges will also flush writes before returning the response to a PIO read.

However, for Jalapeno, the Fire nexus driver, in `ddi_dma_sync()`, will need to do a block read operation to any memory location, followed by a membar sync to ensure that the CPUs internal "invalidate FIFOs" are flushed (since these CPUs allow PIO read returns to pass cache invalidates inside the CPU). This operation is not required for Niagara. Previous generation bridges did not require this operation because they did a DMA to memory to signify the sync was completed. The DMA write does not pass the cache invalidates.

Note – PCIE devices can generate zero byte PCIE reads and/or writes to Fire for data synchronization. Fire will handle these operations without error **as long as** they are to a valid MMU mapped address. **The PCIE specification does not specify how the address for the zero byte operation is determined. This could be an issue if PCIE devices are hardwired to work with PC bridges.**

The latest PCIE Spec (1.0, July 22) says the following things about zero byte reads and writes.

- A Write Request with a length of 1DW with no Bytes enabled is permitted, and has no effect at the Completer.
- If a Read Request of 1 DW specifies that no Bytes are enabled to be read (1st DW BE[3:0] field = 0000b), the corresponding Completion must specify a Length of 1 DW, and include a data payload of 1 DW. The contents of the data payload within the Completion packet is unspecified and may be any value.
- A Memory Read Request of 1 DW with no Bytes enabled, or "zero length Read," may be used by devices as a type of flush Request. For a Requester, the flush semantic allows a device to ensure that previously issued Posted Writes have been completed at their PCI Express Destination.
- The flush semantic has wide application, and all Completers must implement the functionality associated with this semantic. Because a Requester may use the flush semantic without comprehending the characteristics of the Completer, Completers must ensure that zero length reads do not have side-effects. This is really just a specific case of the rule that in a non-prefetchable space, non-enabled Bytes must not be read at the Completer. Note that the flush applies only to traffic in the same Traffic Class as the zero length Read.

1.2.6.8 Event Queue Records

Event queues (EQ) are used to queue interrupts with additional state information. An entry in the event queue is called an event queue record. An event queue record can be written by Fire as a result of a valid PCI Express message received which requires software notification or a MSI/MSI-X being received. More information on what messages generate event queue records and how they are directed to a particular event queue can be found in the following sections.

A MSI/MSI-X as seen on the PCIE bus looks like any other DMA write to Fire. The only way Fire knows that it is a MSI is if it matches a range which is set up by two registers depending upon the DMA is a 64 bit addressed DMA or a 32 bit addressed DMA. These two registers (A side - 0x634000, 0x634008) (B Side - 0x734000, 0x734008) are set up by software to whatever PCIE Address you want. ANY DMA write seen matching in this range will be treated as a MSI/MSI-X and will be sent to the IMU and removed from normal DMA pipeline. So if you have “normal” DMA going to the same address range as is set up to be the range for MSI’s that would be BAD.

Note – This Address matching for MSI/MSI-X CANNOT be disabled. The default address for both the 64 bit and the 32 bit MSI address matching register is an address of all zero’s. Any DMA going to this address even if the necessary steps to set up MSI’s are not taken WILL be removed from the DMA pipeline and passed to the IMU. It is mandatory that no DMA address be sent to this address range if the MSI functionality is not being utilized.

After the MSIs/MSI-Xs makes it to the IMU that PCIE address is no longer used it is thrown on the floor. No matter what it is it will not make it to the MMU. A new address is created from the EQ base address register + EQ number offset plus Tail pointer. Depending on what you set the EQ base address register to determines what the MMU does with the Event Q write. If you set it up as a BYPASS address it will be treated as BYPASS by the MMU if you set it up to be translated it will be translated. This is also up to software to set up.

An event queue record contents include; PCI Express message/Fire Message, Fmt & Type, Packet Length, PCI Express Address[15:2], Requester ID, MSI Data[15:0], msgcode[7:0]. Not all of the fields are valid for a given message. Software can process an event queue record with a single 64-bit read from the event queue record.

All the fields in the EQ Record are from the PCIE header that the MSI/MSI-XMsg came with. Please see the PCIE spec for all of the details about that. These fields are put there unmodified. The length is in DW and for any GOOD MSI/MSI-X it should always be 1 or it will be detected as a malformed MSI.

Table 1-13 Event Queue Record Format (Sheet 1 of 2)

Data Words	Bits	Field	Description
0	63	Reserved	Reserved. Will be set to zero.
0	62:56	FMT/TYPE	1111000 for 64 bit addressed MSI 1011000 for 32 bit addressed MSI 0110xxx for Messages where xxx comply’s with routing rules in PCIE spec
0	55:46	LENGTH	Set to the length of the data.
0	45:32	ADDR[15:2]	Copy of the Address bits 15:2
0	31:16	RID	Requester ID
0	15:00	DATA0	if (FMT/TYPE == 1111000 or 1011000) { DATA0 => MSI DATA [15:0] 15:8 = byte 1 of MSI/MSI-X Data 7:0 = byte 0 of MSI/MSI-X Data } else DATA0 => (8’b0, MSGCODE[7:0])

Table 1-13 Event Queue Record Format (Sheet 2 of 2)

1	63:16	ADDR[63:16]	Copy of the Address bits 63:16
1	15:00	DATA1	if (FMT/TYPE == 1111000 or 1011000) { DATA1 => MSI-X DATA [31:16] (Field NOT valid for MSI) 15:8 = byte 3 of MSI-X Data (Field NOT valid for MSI) 7:0 = byte 2 of MSI-X Data (Field NOT valid for MSI) } else DATA1 => (Copy of Address bits[13:0], 2'b0)
2 - 7	63:00	Reserved	Reserved. Will be set to zero.

1.2.6.8.1 PCI Express Messages

Fire will generate event queue records for the following PCI Express messages; Correctable Error, Uncorrectable Error, Fatal Error, PME, and PME_TO_Ack. Each of these messages can be individually targeted to an event queue using the PCI Express Message Mapping Registers. For example, Uncorrectable Error and Fatal Error may be targeted to event queue #15; and Correctable Error, PME, and PME_TO_Ack, and may be targeted to event queue #14. As an example for this case, PME messages will then always generate event queue records in event queue #14.

1.2.6.8.2 Message Signaled Interrupts (MSIs and MSI-Xs)

MSIs and MSI-Xs are inband posted writes to a pre-programmed address which can be either a 32-bit or a 64-bit addressed. MSIs and MSI-Xs allow up to 32 unique interrupts to be generated from each function within a device. Fire will check either the 32 bit or the 64 bit address on a PCI Express write to determine if it is a MSI or an MSI-X. If the incoming write is 64 bit addressed the MSI 64-bit Address register (0x00634008, 0x00734008) will be used for the comparison. If the incoming write is 32 bit addressed the MSI 32-bit Address register (0x00634000, 0x00734000) will be used for the comparison. The address used to determine if the write is a MSI or MSI-X is the same. Two separate address CANNOT be specified to distinguish MSIs from MSI-Xs.

A 16 bit "Message Data" field is transmitted as part of the MSI. A 32 bit "Message Data" field is transmitted as part of the MSI-X. The contents of this field are programmable and are located in the function's PCI/PCI Express configuration space (capability list).

If a device function reports that it is Multiple Message Capable, it will also report the maximum number of unique MSIs or MSI-Xs that it can generate (from 1 to 32 in powers of 2). System software can enable all or a subset of the MSIs (from 1 to MAX in powers of 2). If a device function is configured to generate multiple unique MSIs or MSI-Xs, the device will use the lower data bits in the MSI 16-bit data or the lower data bits in the MSI-X32-bit field to signify the interrupt number (from 0 to MAX-1). The number of bits used is N where $2^N = \text{MAX}$. The lower N bits of the "Message Data" field located in the function's PCI configuration space are ignored for this case.

Fire implements a MSI/MSI-X mapping table, located within Fire, to provide interrupt distribution and interrupt coalescing (for MSIs/MSI-Xs which do not require interrupts to be acknowledged by software). Since PCI Express does not allow us to set an individual 16-bit data field for each unique MSI or the individual 32-bit data field for

each unique MSI-X that the device function supports, the MSI/MSI-X mapping table will map a MSI/MSI-X to an event queue. It also will prevent an interrupt being written multiple times into an event queue. This prevents the device from overflowing the event queue with the same interrupt. The number of MSIs/MSI-Xs supported by Fire is limited by the size of the MSI/MSI-X mapping table. Fire's mapping table supports 256 MSIs/MSI-Xs, therefore Fire supports up to 256 MSIs/MSI-Xs. The MSI/MSI-X mapping table is indexed into using the lower 8 bits of the 16-bit MSI data field or the lower 8 bits of the 32-bit MSI-X data field.

A device function's MSIs/MSI-Xs must be mapped both aligned and contiguous within the MSI mapping table (i.e. if the function supports four interrupts, those four interrupts must take up four contiguous entries and be aligned on a four entry boundary in the MSI/MSI-X Mapping Table).

The MSI/MSI-X mapping state includes; valid/invalid entry, OK to write to EQ, and the EQ number to write Event Record to.

Note – Before processing an MSI, software must set the “OK to write to EQ” bit in the MSI state and then readback the MSI mapping register to ensure the write completed.

1.2.7 EStar Mode

Fire Supports JBus's Reduced Power Mode (Estar). Power is reduced by running the JBus logic (and logic clocked by the JBus clock, in Fire this includes the DMC logic as well as JBC logic) at reduced clock frequencies. The JBus clock can either be full frequency, half speed mode or 1/32 speed mode.

Note that the boot path Fire is the Estar mode controller (i.e. initiates a J_CHNG_L sequence on JBus). All devices must be configured to be in the same speed mode before the boot path Fire is commanded to initiate a JBus speed change sequence.

In addition to the JBus speed change registers in the JBC, there is also a register in the ILU (ILU Device Capabilities register). For system that will support Estar mode (regardless of the running speed) this register must be initialized to be in system Estar supported mode before any traffic to PCI-E ports.

See the operational section for more detail on the programming of the Estar control registers.

1.2.8 PCI Power Management

Fire supports PCI Express link active state power management state L0s and L1. Both of these behaviors can be enabled/disabled by software and default to disabled.

Fire does not support the PCI Express link power management state L2, since it does not support being powered from Auxiliary power. Therefore, Fire cannot be used to provide features such as Wake On Lan (WOL).

Fire supports entering the PCI Express link power management state L1 on an upstream port due to a request from the downstream device attached to that port. This behavior can be enabled/disabled by software and defaults to disabled. Fire will never generate a request downstream to enter L1 state.

There are four power management message in PCI Express. They are:

- PM_Active_State_NAK
- PME_Turn_Off
- PM_PME
- PME_TO_Ack

Fire will silently handle PM_Active_State_NAK messages. There is no software visibility for these messages.

The PME_Turn_Off message will be generated by Fire when software writes the PTO bit in the TLU PME Turn Off Generate Register.

The PM_PME and PME_TO_Ack messages will be written to programmable event queue(s) by Fire. Fire will do no further processing on behalf of these messages. It is up to software to handle these messages (i.e. time-out a PME_TO_Ack message in response to a PME_Turn_Off message).

The following describes the sequence required to transition from PCI Express link power management state L0 to PCI Express link power management state L2/L3 Ready.

Software directs all functions of a downstream component to D3(hot). The downstream component then initiates the transition of the link to L1 as required by the PCI Express specification. Software then waits for the link to transition into L1 (reading the LPU LTSSM Status1 Register to determine the link state). If the link transitions to L1, software then causes Fire to broadcast the PM_Turn_Off message by writing to the PTO bit in the PME Turn Off Generate Register. This message causes the Link to transition back to L0 in order to send the PM_Turn_Off message, and to enable the downstream component to respond with PM_TO_Ack. The PM_TO_Ack message is placed in an Event Queue by Fire for software to process. It is up to software to time-out if a PME_TO_Ack is not received. After the PM_TO_Ack is sent, the downstream component then initiates the L2/L3 Ready transition protocol. Software then waits for the link to transition into L2/L3 Ready, which ends in L1 state due to Fire's specific implementation (reading the LPU LTSSM Status1 Register to determine the link state). If the link transitions to L1 state, software can then power off the link.

Handling of the failure to transition to L1 or L2/L3 Ready is software implementation specific.

1.2.9 *Endianess.*

This section assumes the reader already has a basic understanding of Endianess.

Since PCI Express is a serial bus, Fire does not need to worry about the "Byte Twisting" done on previous generations of Sun's PCI bridges. Fire ensures that the byte stream order is maintained (e.g. the first valid byte on JBus is put into data byte 0 in the PCI Express packet, etc.).

What should be relatively simple, gets a little confusing when we overload some of the terms. JBus is a 128-bit (16 byte) big endian bus. Therefore, byte 0 is the most significant byte (MSB) and byte 15 is the least significant byte (LSB). There is a 16-bit byte mask to specify the valid bytes during the transfer. Following SPARC conventions, bit numbering is little endian. Therefore, bit 0 is the least significant bit and is the byte select for byte 0 (which is the MSB), bit 15 is the most significant bit and is the byte select for byte 15 (which is the LSB). It looks a little strange the first time you put it down on paper. To continue, byte 0 on JBus does not necessarily map to data byte 0 in the PCI-E packet. The first valid byte on JBus, which could be byte #8 (for example), will be data byte 0 in the PCI-E packet.

PCI Express packet headers are big endian. PCI Express data is endian neutral (i.e. a byte stream). For places where a portion of the PCI Express header is place into memory (e.g. event queue records), the data will maintain big endian structure (i.e. Fire will never swap bytes around).

Endianess is created based on how structured data is placed into the byte stream. How a device organizes its registers and descriptors is device specific. PCI configuration space is structured as little endian. Therefore, when a MSI is sent by a device, 16-bit MSI data word will be little endian structured.

The same semantics which worked for Psycho/Schizo, work with Fire. If a device is a little endian device, expect DMA data to be little endian structured. If a device is a big endian device, expect DMA data to be big endian structured.

The one exception will be Event Queue records for MSIs. The header data will be big endian, and the MSI data will be little endian in the event queue record.

1.2.10 Error Register Overview

For **each** error bit, there are five required registers implemented in the Fire ASIC (excluding the LPU). An optional sixth register may be implemented for some error types to log any helpful additional information. This register may be shared between multiple, but similar errors. The registers are:

- **Interrupt Status Register** - Indicates whether the particular error is set and enabled for interrupt. SW will read this register to determine the error interrupt source.
- **Error Status Clear Register** - Used by SW to clear the error. Can also be read by SW to determine if the error is set, regardless of whether it will cause an interrupt (for the case when interrupts are not enabled).
- **Interrupt Enable Register** - Used by SW to enable a particular error for interrupt.
- **Error Log Enable Register** - Used by SW to enable a particular error for logging. Errors must be enabled for logging in order for them to be enabled for and cause an interrupt.
- **Error Status Set Register** - Used by SW to force a particular error. If interrupts are enabled for the particular error, then setting this register will cause an interrupt. Diagnostic use only.
- **Error Log Register** - This is an optional register that is used to hold any necessary additional information that is associated with a particular error. Such information might be transaction address, id, command type, etc.

For each error bit Fire maintains **Primary** and **Secondary** error register bits, used as follows; on the first occurrence of an error the Primary bit is set. If enabled an interrupt will occur and software will handle and eventually clear the error status from the regs. If the **same** error were to occur again before software cleared the Primary event, then the Secondary bit will be set and if enabled an interrupt will occur. If the **same** error were to occur after hardware handled and cleared the Primary condition then of course the Primary bit will again get set rather than the Secondary.

Programming Note:

The Secondary Error Interrupt should not be enabled without the Primary Error Interrupt enabled. Hardware will still function properly only reporting the secondary error (should one occur) but software has the potential to miss Primary errors by programming in this manner.

Programming Note:

Error Log Registers are freely loaded, only holding data should an error occur (e.g. the log register stops loading and is 'locked'). Once an error has occurred, the log register can be read for relevant data. Writing to the corresponding Error Status Clear register will clear the error and re-enable the log register to start clocking data once again.

Errors may be grouped together where errors share a common log register because they require the same additional information. In this case only information associated with the primary error is captured. Should any secondary errors or other primary errors in the same group occur, their information will not be captured since the log register is already locked by the first primary error.

Error reporting mechanism in LPU is slightly different. Below is a summary table to map Sun's error register to LPU's.

Table 1-14 Mapping between Sun's and LPU's error reporting registers

Sun's Error Register	LPU's Error Register
Error Log Enable	No corresponding register However, error log is always enabled
Interrupt Enable	Interrupt Mask (Note: mask bit value 1 represents interrupt disabled and 0 represents interrupt enabled)
Interrupt Status	LPU top level Interrupt Status Register & bit [31] of Interrupt and Status Register
Error Status Clear	Bits [30:0] of Interrupt and Status Register
Error Status Set	Interrupt and Status Test Note: reads of the register return zero.

For further details on the error registers refer to:

- The specific register descriptions for additional details on each of them.
- Error summary table at the end of this spec.

1.2.11 Performance Register Overview

Fire provides 20 performance counter registers throughout the chip to aid in analysis, debug, and performance tuning applications. The registers track events in the link layer, transaction layer, MMU, interrupt unit, and JBus areas.

Fire Performance Counter Registers

Table 1-15 Fire Performance Counter Registers

Location (block)	Register	Size (bits)	Clock Domain
JBC	JBC Performance Counter Zero Register	64	JBUS
JBC	JBC Performance Counter One Register	64	JBUS
PCI-A	IMU Performance Counter Zero Register	64	JBUS
PCI-A	IMU Performance Counter One Register	64	JBUS
PCI-A	MMU Performance Counter Zero Register	64	JBUS
PCI-A	MMU Performance Counter One Register	64	JBUS
PCI-A	TLU Performance Counter Zero Register	64	PCIE
PCI-A	TLU Performance Counter One Register	64	PCIE
PCI-A	TLU Performance Counter Two Register	32	PCIE
PCI-A	LPU Link Performance Counter1	32	PCIE
PCI-A	LPU Link Performance Counter2	32	PCIE
PCI-B	IMU Performance Counter Zero Register	64	JBUS
PCI-B	IMU Performance Counter One Register	64	JBUS
PCI-B	MMU Performance Counter Zero Register	64	JBUS
PCI-B	MMU Performance Counter One Register	64	JBUS
PCI-B	TLU Performance Counter Zero Register	64	PCIE
PCI-B	TLU Performance Counter One Register	64	PCIE
PCI-B	TLU Performance Counter Two Register	32	PCIE
PCI-B	LPU Link Performance Counter1	32	PCIE
PCI-B	LPU Link Performance Counter2	32	PCIE

Each register group contains a corresponding Select register used to select the specific input to the counter and in the case of the LPU counters, there are additional counter control and test registers.

Programming Note:

Counting does not occur when the select is set to zero. Counters are cleared by SW when writing them to zero (or by reset) and not by transitioning the select. If the select is transitioned between various events without resetting the counters an aggregate value will be contained in the counters (i.e. the counters keep on counting without returning to zero).

See the specific register descriptions for additional details on the counters.

1.2.12 Link Layer Thresholds

There are three types of thresholds in link layer, which are

- Ack Latency Timer Threshold - used for determining a Ack DLLP's priority
- Frequent Nak Latency Timer Threshold - used to resend a Nak DLLP
- Replay Timer Threshold - used to replay TLPs

1.2.12.1 Ack Latency Timer Threshold

The default value in LPU Txlink Ack Latency Timer Threshold Register for Fire is 0x5 in symbol times. In order that Acks can be sent out within the Ack latency transmission limit, it is recommended to program a low value in this register (but not lower than 0x4).

1.2.12.2 Frequent Nak Latency Timer Threshold

The value to program into LPU Txlink Frequent Nak Latency Timer Threshold Register is a function of the link operating width, max payload size, and Tx_L0s adjustment factor when transmit L0s is enabled.

Table 1-16 Frequent Nak Latency Threshold Values without Tx_L0s adjustment factor

Max Payload Size	Link Operating Width			
	X1	X4	X8	X16
128B	0xED (237dec)	0x49 (73 dec)	0x43 (67 dec)	0x30 (48 dec)
256B	0x1A0 (416dec)	0x76 (118 dec)	0x6B (107 dec)	0x48 (72 dec)
512B	0x22F (559dec)	0x9A (154 dec)	0x56 (86 dec)	0x56 (86 dec)
1024B	0x42F (1071dec)	0x11A (282 dec)	0x96 (150 dec)	0x96 (150 dec)
2048B	0x82F (2095dec)	0x21A (538 dec)	0x116 (278 dec)	0x116 (278 dec)
4096B	0x102F (4143dec)	0x41A (1050 dec)	0x216 (534 dec)	0x216 (534 dec)

If L0s is enabled, the Tx_L0s adjustment factor needs to be added to the Frequent Nak Latency timer threshold values in the table before programming the actual value, where

If Extended Sync =0 then:

$$\text{Tx_L0s adjustment factor} = (\text{Received FTS number} + 1) * 4$$

If Extended Sync = 1, then:

$$\text{Tx_L0s adjustment factor} = (4096+1) * 4$$

The default Tx_L0s adjustment factor is 0. This value is derived from the Received FTS Number [15:8] in Receive Phy Status 2 register.

Note: the threshold values indicated in the table might change depending on actual internal processing delays for received TLPs and transmitted DLLPs.

1.2.12.3 Replay Timer Threshold

The value to program into LPU Txlink Replay Timer Threshold Register is a function of the link operating width, max payload size, and Rx_L0s adjustment factor.

The values in table below are taken from PCI-E spec plus a margin of 25%, in the case of without Rx_L0s adjustment factor. The 25% margin is considered as taking the internal delay into account.

Table 1-17 Replay Timer threshold values without Rx_L0s adjustment factor, using a margin of 25%

Max Payload Size	Link Operating width			
	X1	X4	X8	X16
128B	0x379 (889 dec)	0x112 (274 dec)	0xFC (252 dec)	0xB4 (180 dec)
256B	0x618 (1560 dec)	0x1BA (442 dec)	0x192 (402 dec)	0x10E (270 dec)
512B	0x831 (2097 dec)	0x242 (578 dec)	0x143 (323 dec)	0x143 (323 dec)
1024B	0xFB1 (4017 dec)	0x422 (1058 dec)	0x233 (563 dec)	0x233 (563 dec)
2048B	0x1EB0 (7856dec)	0x7E1(2017 dec)	0x412 (1042 dec)	0x412 (1042 dec)
4096B	0x3CB0 (15536dec)	0xF61 (3973dec)	0x7D2 (2002 dec)	0x7D2 (2002 dec)

The Rx_L0s adjustment factor needs to be added to the Replay timer threshold values in the table above before programming the value in LPU Txlink Replay Timer Threshold Register, where

$$\text{Rx_L0s adjustment factor} = (\text{Transmit FTS number} + 1) * 4$$

The default Rx_L0s adjustment factor is 0x234 or 564 decimal ((0x8C+1)*4). This value is derived from the Transmit FTS Number[15:8] in LTSSM Configuration 4 register.

Note: the threshold values indicated in the table above might change depending on actual internal processing delays for received TLPs and transmitted DLLPs.

1.2.13 Link Ingress Initial Posted Data Credit

If a processor can issue 16 outstanding PIO block (cache line - 64-byte) reads, the following SW work around must be applied. The value of “Posted Data Credit” in “TLU Ingress Credits Initial Register” must be programmed by SW to avoid a potential ingress posted data RAM overflow. The default value is 0xC0, but the allowed max. value is 0xBC. See Fire Chip Errata number 2.0-6 for detailed information.

In Niagara or Jalapeno systems, since Niagara issues up to 16 outstanding PIO read with length up to 16 bytes, Jalapeno only issues one outstanding PIO read at a time, **no** SW work around is needed to reduce the initial “Posted Data Credit” from its default value.

1.2.14 Register Reset Behavior and SW Access to Fire

There are two reset signals internal to Fire and referenced in the Reset Value field of all register descriptions contained in Section 1.3.

Table 1-18 Fire Reset Value Definitions

Reset	Comment
rst_l	All registers reset except Error log/status registers, Reset Source register,... (see actual register descriptions)
por_l	All registers reset (including those reset by rst_l)

In general terms, rst_l is considered a soft reset, as most registers are reset by it, but some are not and instead they maintain their state across the reset activity. In contrast, por_l is considered a hard reset since all registers in Fire are effected by it.

Note that upon power up, reset is not effective in Fire or on its output pins until the F_PWR_GOOD input signal is asserted and the internal clocks have sampled it for 10 JBus clock cycles.

Programming Note:

Note that resetting Fire, regardless of reset type, will cause the base/mask registers to reset, thus removing any prior mapping to respective base/mask address spaces. However access to Fire PIO space itself, including the internal PIO ring, by software is unaffected by reset as it is hardwired and based on JBus AgentID and address.

In programming the base/mask registers, order is un-important, however software access to the respective base/mask address space should not be attempted until both registers are properly programmed.

1.3 CSR Fields and Bits

1.3.1 General Information

Fire's Configuration and Status Registers (CSRs) are described in this chapter. The following abbreviations are used throughout this chapter:

- R - Read only: a register field that is not writable
- RW - Read/write: A standard register field
- RW1C - Read/write 1 to clear: Writing a 0 to bits in this field has no affect, but writing a 1 to a bit in this field will cause that bit to be set to 0.
- RW1S - Read/write 1 to set: Writing a 0 to bits in this field has no affect, but writing a 1 to a bit in this field will cause that bit to be set to 1.
- L - Write only: reads of this field return 0.
- DMA - Direct Memory Access: A transaction initiated by a leaf block resulting in a JBus transaction
- DVMA - Direct Virtual Memory Access: A subclass of DMA transactions in which the address of the leaf transaction is treated as a virtual address and translated by an MMU.

Note that Reserved bits read as 0.

1.3.1.1 Access Size

All registers in Fire have a natural access size of 8 bytes. Register accesses must always be done with the appropriate access size, or undefined behavior will result.

1.3.1.2 Unimplemented Addresses

Any address within Fire's register address space that is not documented here as belonging to a specific register is reserved and should not be read or written by software. Although, Fire will "own" and respond to its entire 128 Gbyte address range, which is the allocated space for each agent attached to JBus (64 Gbyte of cacheable space, and 64 Gbyte of non-cacheable space), a read to an unimplemented address can return a Read Error or undefined data and a write will be ignored. Since Fire will cover its entire allocated address space, system hangs due to accesses to unimplemented addresses will be reduced, although this behavior should still be avoided by software.

1.3.1.3 Physical Addresses

The 43-bit base physical address for Fire's 8 MByte noncacheable configuration region is:

$\{1, 0, 13'b0, \text{AgentID}[4:0], 23'b0\}$

where AgentID[4:0] is Fire's JBus Port ID.

All register addresses are documented as offsets within Fire's 8MByte noncacheable configuration region. For more details about Fire's address regions, see Section 1.2.2.1, "JBUS Physical Address Partitioning," on page 1-23.

1.3.2 Register Map

The following table summarize all of the on chip registers within Fire. All accesses to these registers must be 8 byte, 8 byte aligned except for access to the Device ID register, which sits in the Fcode region (0x00.0000 - 0x3F.FFF8).

Table 1-19 Fire Register Map (Sheet 1 of 19)

Offset	Register
JBus CSR Space	
0x000000	JBus Device ID Register
0x000008-0x400018	RESERVED
0x400020	EBus Offset Base Register
0x400028	EBus Offset Mask Register
0x400030-0x400038	RESERVED
0x400040	PCIE-A Mem32 Offset Base Register
0x400048	PCIE-A Mem32 Offset Mask Register
0x400050	PCIE-A Cfg/IO Offset Base Register
0x400058	PCIE-A Cfg/IO Offset Mask Register
0x400060	PCIE-B Mem32 Offset Base Register
0x400068	PCIE-B Mem32 Offset Mask Register
0x400070	PCIE-B Cfg/IO Offset Base Register
0x400078	PCIE-B Cfg/IO Offset Mask Register
0x400080	PCIE-A Mem64 Offset Base Register
0x400088	PCIE-A Mem64 Offset Mask Register
0x400090	PCIE-B Mem64 Offset Base Register
0x400098	PCIE-B Mem64 Offset Mask Register
0x4000A0-0x40FFF8	RESERVED
0x410000	Fire Control/Status Register
0x410008-0x410048	RESERVED
0x410050	JBus PLL Control and DTL Control Register
0x410058	JBus Energy Star Control Register
0x410060	JBus Change Initiation Control Register
0x410068-0x417008	RESERVED
0x417010	Reset Generation Register
0x417018	Reset Source Register
0x417020-0x45FFF8	RESERVED
0x460000	GPIO Port 0 Pin 0 Data Register
0x460008	GPIO Port 0 Pin 1 Data Register
0x460010	GPIO Port 0 Pin 2 Data Register
0x460018	GPIO Port 0 Pin 3 Data Register

Table 1-19 Fire Register Map (Sheet 2 of 19)

Offset	Register
0x460020	GPIO Port 0 Data Register
0x460028	GPIO Port 0 Control Register
0x460030-0x461FF8	RESERVED
0x462000	GPIO Port 1 Pin 0 Data Register
0x462008	GPIO Port 1 Pin 1 Data Register
0x462010	GPIO Port 1 Pin 2 Data Register
0x462018	GPIO Port 1 Pin 3 Data Register
0x462020	GPIO Port 1 Data Register
0x462028	GPIO Port 1 Control Register
0x462030-0x463FF8	RESERVED
0x464000	EBus EPROM Timing Control Register
0x464008	EBus Chip Select 1 Timing Control Register
0x464010	EBus Chip Select 2 Timing Control Register
0x464018	EBus Chip Select 3 Timing Control Register
0x464020-0x465FF8	RESERVED
0x466000	I2C 0 Input Monitor Register
0x466008	I2C 0 Data Drive Register
0x466010	I2C 0 Clock Drive Register
0x466018-0x467FF8	RESERVED
0x468000	I2C 1 Input Monitor Register
0x468008	I2C 1 Data Drive Register
0x468010	I2C 1 Clock Drive Register
0x468018-0x46FFF8	RESERVED
0x470000	PCIE-A Leaf CSR Ring Slow Only Access
0x470008	PCIE-B Leaf CSR Ring Slow Only Access
0x470010	JBus Parity Control Register
0x470018	JBus Scratch Register 1
0x470020	JBus Scratch Register 2
0x470028	JBC Error Logic Analyzer Trigger Enable for J_ERR
0x470030	JBus Scratch Persistent Register
0x470038-0x470FF8	RESERVED
0x471000	JBC Error Log Enable Register
0x471008	JBC Interrupt Enable Register
0x471010	JBC Interrupt Status Register
0x471018	JBC Error Status Clear Register
0x471020	JBC Error Status Set Register

Table 1-19 Fire Register Map (Sheet 3 of 19)

Offset	Register
0x471028	JBC Fatal Reset Enable Register
0x471030	JBCINT In Transaction Error Log Register
0x471038	JBCINT In Transaction Error Log Register 2
0x471040	JBCINT Out Transaction Error Log Register
0x471048	JBCINT Out Transaction Error Log Register 2
0x471050	Fatal Error Log Register 1
0x471058	Fatal Error Log Register 2
0x471060	Merge Transaction Error Log Register
0x471068	DMCINT ODCD Error Log Register
0x471070	DMCINT IDC Error Log Register
0x471078	CSR Error Log Register
0x471080-0x4717F8	RESERVED
0x471800	JBC Core and Block Interrupt Enable Register
0x471808	JBC Core and Block Error Status Register
0x471810-0x471FF8	RESERVED
0x472000	JBC Performance Counter Select Register
0x472008	JBC Performance Counter Zero Register
0x472010	JBC Performance Counter One Register
0x472018-0x472FF8	RESERVED
0x473000	Fire and JBC Debug Select Register A
0x473008	Fire and JBC Debug Select Register B
0x473010-0x51FFF8	RESERVED
0x520000	I2C Bus-x Slave Address Register
0x520008	I2C Bus-x Extended Slave Address Register
0x520010	I2C Bus-x Data Byte Register
0x520018	I2C Bus-x Control Register
0x520020	I2C Bus-x Status Register
0x520028	I2C Bus-x Clock Control Register
0x520030	I2C Bus-x Software Reset Register
0x520038-0x52FFF8	RESERVED
0x530000	I2C Bus-x Slave Address Register
0x530008	I2C Bus-x Extended Slave Address Register
0x530010	I2C Bus-x Data Byte Register
0x530018	I2C Bus-x Control Register
0x530020	I2C Bus-x Status Register
0x530028	I2C Bus-x Clock Control Register

Table 1-19 Fire Register Map (Sheet 4 of 19)

Offset	Register
0x530030	I2C Bus-x Software Reset Register
0x530038-0x5FFFF8	RESERVED
PCIE-A CSR Space	
0x600000-0x600FF8	RESERVED
0x601000-0x6011F8	Interrupt Mapping Registers
0x601200-0x6013F8	RESERVED
0x601400-0x6015F8	Interrupt Clear Registers
0x601600-0x6019F8	RESERVED
0x601A00	Interrupt Retry Timer Register
0x601A08	RESERVED
0x601A10	Interrupt State Status Register 1
0x601A18	Interrupt State Status Register 2
0x601A20-0x60AFF8	RESERVED
0x60B000	INTX Status Register
0x60B008	INT A Clear Register
0x60B010	INT B Clear Register
0x60B018	INT C Clear Register
0x60B020	INT D Clear Register
0x60B028-0x60FFF8	RESERVED
0x610000	Event Queue Base Address Register
0x610008-0x610FF8	RESERVED
0x611000-0x611118	Event Queue Control Set Register
0x611120-0x6111F8	RESERVED
0x611200-0x611318	Event Queue Control Clear Register
0x611320-0x6113F8	RESERVED
0x611400-0x611518	Event Queue State Register
0x611520-0x6115F8	RESERVED
0x611600-0x611718	Event Queue Tail Register
0x611720-0x6117F8	RESERVED
0x611800-0x611918	Event Queue Head Register
0x611920-0x61FFF8	RESERVED
0x620000-0x6207F8	MSI Mapping Register
0x620800-0x627FF8	RESERVED
0x628000-0x6287F8	MSI Clear Registers
0x628800-0x62BFF8	RESERVED
0x62C000	Interrupt Mondo Data 0 Register

Table 1-19 Fire Register Map (Sheet 5 of 19)

Offset	Register
0x62C008	Interrupt Mondo Data 1 Register
0x62C010-0x62FFF8	RESERVED
0x630000	ERR COR Mapping Register
0x630008	ERR NONFATAL Mapping Register
0x630010	ERR FATAL Mapping Register
0x630018	PM PME Mapping Register
0x630020	PME To ACK Mapping Register
0x630028-0x630FF8	RESERVED
0x631000	IMU Error Log Enable Register
0x631008	IMU Interrupt Enable Register
0x631010	IMU Interrupt Status Register
0x631018	IMU Error Status Clear Register
0x631020	IMU Error Status Set Register
0x631028	IMU RDS Error Log Register
0x631030	IMU SCS Error Log Register
0x631038	IMU EQS Error Log Register
0x631040-0x6317F8	RESERVED
0x631800	DMC Core and Block Interrupt Enable Register
0x631808	DMC Core and Block Error Status Register
0x631810	Multi Core Error Status Register
0x631818-0x631FF8	RESERVED
0x632000	IMU Performance Counter Select Register
0x632008	IMU Performance Counter Zero Register
0x632010	IMU Performance Counter One Register
0x632018-0x633FF8	RESERVED
0x634000	MSI 32-bit Address Register
0x634008	MSI 64-bit Address Register
0x634010	RESERVED
0x634018	Mem 64 PCIE Offset Register
0x634020-0x63FFF8	RESERVED
0x640000	MMU Control and Status Register
0x640008	MMU TSB Control Register
0x640010-0x6400F8	RESERVED
0x640100	MMU TTE Cache Flush Address Register
0x640108	MMU TTE Cache Invalidate Register
0x640110-0x640FF8	RESERVED

Table 1-19 Fire Register Map (Sheet 6 of 19)

Offset	Register
0x641000	MMU Error Log Enable Register
0x641008	MMU Interrupt Enable Register
0x641010	MMU Interrupt Status Register
0x641018	MMU Error Status Clear Register
0x641020	MMU Error Status Set Register
0x641028	MMU Translation Fault Address Register
0x641030	MMU Translation Fault Status Register
0x641038-0x641FF8	RESERVED
0x642000	MMU Performance Counter Select Register
0x642008	MMU Performance Counter Zero Register
0x642010	MMU Performance Counter One Register
0x642018-0x645FF8	RESERVED
0x646000-0x6461F8	MMU TTE Cache Virtual Tag Registers
0x646200-0x646FF8	RESERVED
0x647000-0x6471F8	MMU TTE Cache Physical Tag Registers
0x647200-0x647FF8	RESERVED
0x648000-0x648FF8	MMU TTE Cache Data Registers
0x649000-0x650FF8	RESERVED
0x651000	ILU Error Log Enable Register
0x651008	ILU Interrupt Enable Register
0x651010	ILU Interrupt Status Register
0x651018	ILU Error Status Clear Register
0x651020	ILU Error Status Set Register
0x651028-0x6517F8	RESERVED
0x651800	PEC Core and Block Interrupt Enable Register
0x651808	PEC Core and Block Interrupt Status Register
0x651810-0x651FF8	RESERVED
0x652000	ILU Device Capabilities Register
0x652008-0x652FF8	RESERVED
0x653000	DMC Debug Select Register for Port A
0x653008	DMC Debug Select Register for Port B
0x653010-0x6530F8	RESERVED
0x653100	DMC PCI Express Configuration Register
0x653108-0x65FFF8	RESERVED
0x660000-0x6600F8	Packet Scoreboard DMA Register Set
0x660100-0x663FF8	RESERVED

Table 1-19 Fire Register Map (Sheet 7 of 19)

Offset	Register
0x664000-0x664078	Packet Scoreboard PIO Register Set
0x664080-0x66FFF8	RESERVED
0x670000-0x6700F8	Transaction Scoreboard Register Set
0x670100	Transaction Scoreboard Status Register
0x670108-0x67FFF8	RESERVED
0x680000	TLU Control Register
0x680008	TLU Status Register
0x680010	TLU PME Turn Off Generate Register
0x680018	TLU Ingress Credits Initial Register
0x680020-0x6800F8	RESERVED
0x680100	TLU Diagnostic Register
0x680108-0x6801F8	RESERVED
0x680200	TLU Egress Credits Consumed Register
0x680208	TLU Egress Credit Limit Register
0x680210	TLU Egress Retry Buffer Register
0x680218	TLU Ingress Credits Allocated Register
0x680220	TLU Ingress Credits Received Register
0x680228-0x680FF8	RESERVED
0x681000	TLU Other Event Log Enable Register
0x681008	TLU Other Event Interrupt Enable Register
0x681010	TLU Other Event Interrupt Status Register
0x681018	TLU Other Event Status Clear Register
0x681020	TLU Other Event Status Set Register
0x681028	TLU Receive Other Event Header1 Log Register
0x681030	TLU Receive Other Event Header2 Log Register
0x681038	TLU Transmit Other Event Header1 Log Register
0x681040	TLU Transmit Other Event Header2 Log Register
0x681048-0x681FF8	RESERVED
0x682000	TLU Performance Counter Select Register
0x682008	TLU Performance Counter Zero Register
0x682010	TLU Performance Counter One Register
0x682018	TLU Performance Counter Two Register
0x682020-0x682FF8	RESERVED
0x683000	TLU Debug Select A Register
0x683008	TLU Debug Select B Register
0x683010-0x68FFF8	RESERVED

Table 1-19 Fire Register Map (Sheet 8 of 19)

Offset	Register
0x690000	TLU Device Capabilities Register
0x690008	TLU Device Control Register
0x690010	TLU Device Status Register
0x690018	TLU Link Capabilities Register
0x690020	TLU Link Control Register
0x690028	TLU Link Status Register
0x690030	TLU Slot Capabilities Register
0x690038-0x690FF8	RESERVED
0x691000	TLU Uncorrectable Error Log Enable Register
0x691008	TLU Uncorrectable Error Interrupt Enable Register
0x691010	TLU Uncorrectable Error Interrupt Status Register
0x691018	TLU Uncorrectable Error Status Clear Register
0x691020	TLU Uncorrectable Error Status Set Register
0x691028	TLU Receive Uncorrectable Error Header1 Log Register
0x691030	TLU Receive Uncorrectable Error Header2 Log Register
0x691038	TLU Transmit Uncorrectable Error Header1 Log Register
0x691040	TLU Transmit Uncorrectable Error Header2 Log Register
0x691048-0x6A0FF8	RESERVED
0x6A1000	TLU Correctable Error Log Enable Register
0x6A1008	TLU Correctable Error Interrupt Enable Register
0x6A1010	TLU Correctable Error Interrupt Status Register
0x6A1018	TLU Correctable Error Status Clear Register
0x6A1020	TLU Correctable Error Status Set Register
0x6A1028-0x6E1FF8	RESERVED
0x6E2000	LPU ID Register
0x6E2008	LPU Reset Register
0x6E2010	LPU Debug Status Register
0x6E2018	LPU Debug Config Register
0x6E2020	LPU LTSSM Control Register
0x6E2028	LPU Link Status Register
0x6E2030-0x6E2038	RESERVED
0x6E2040	LPU Interrupt Status Register
0x6E2048	LPU Interrupt Mask Register
0x6E2050-0x6E20F8	RESERVED
0x6E2100	LPU Link Performance Counter Select Register
0x6E2108	RESERVED

Table 1-19 Fire Register Map (Sheet 9 of 19)

Offset	Register
0x6E2110	LPU Link Performance Counter Control Register
0x6E2118	RESERVED
0x6E2120	LPU Link Performance Counter1
0x6E2128	LPU Link Performance Counter1 Test
0x6E2130	LPU Link Performance Counter2
0x6E2138	LPU Link Performance Counter2 Test
0x6E2140-0x6E21F8	RESERVED
0x6E2200	LPU Link Layer Config Register
0x6E2208	LPU Link Layer Status Register
0x6E2210	LPU Link Layer Interrupt and Status Register
0x6E2218	LPU Link Layer Interrupt and Status Test Register
0x6E2220	LPU Link Layer Interrupt Mask Register
0x6E2228-0x6E2238	RESERVED
0x6E2240	LPU Flow Control Update Control Register
0x6E2248-0x6E2258	RESERVED
0x6E2260	LPU Link Layer Flow Control Update Timeout Value Register
0x6E2268	LPU Link Layer VC0 Flow Control Update Timer0 Register
0x6E2270	LPU Link Layer VC0 Flow Control Update Timer1 Register
0x6E2278-0x6E23F8	RESERVED
0x6E2400	LPU Txlink Frequent Nak Latency Timer Threshold Register
0x6E2408	LPU Txlink AckNak Latency Timer Register
0x6E2410	LPU Txlink Replay Timer Threshold Register
0x6E2418	LPU Txlink Replay Timer Register
0x6E2420	LPU Txlink Replay Number Status Register
0x6E2428	LPU Replay Buffer Max Address Register
0x6E2430	LPU Txlink Retry FIFO Pointer Register
0x6E2438	LPU Txlink Retry FIFO R/W Pointer Register
0x6E2440	LPU Txlink Retry FIFO Credit Register
0x6E2448	LPU Txlink Sequence Counter Register
0x6E2450	LPU Txlink Ack Sent Sequence Number Register
0x6E2458	LPU Txlink Sequence Count FIFO Max Addr Register
0x6E2460	LPU Txlink Sequence Count FIFO Pointers Register
0x6E2468	LPU Txlink Sequence Count R/W Pointers Register
0x6E2470	LPU Txlink Test Control Register
0x6E2478	RESERVED
0x6E2480	LPU Txlink Memory Address Control Register

Table 1-19 Fire Register Map (Sheet 10 of 19)

Offset	Register
0x6E2488	LPU Txlink Memory Data Load0 Register
0x6E2490	LPU Txlink Memory Data Load1 Register
0x6E2498	LPU Txlink Memory Data Load2 Register
0x6E24A0	LPU Txlink Memory Data Load3 Register
0x6E24A8	LPU Txlink Memory Data Load4 Register
0x6E24B0-0x6E24B8	RESERVED
0x6E24C0	LPU Txlink Retry Data Count Register
0x6E24C8	LPU Txlink Sequence Buffer Count Register
0x6E24D0	LPU Txlink Sequence Buffer Bottom Data Register
0x6E24D8	RESERVED
0x6E24E0	LPU Txlink Ack Latency Timer Threshold Register
0x6E24E8-0x6E24F8	RESERVED
0x6E2500	LPU Rxlink Next Receive Sequence + 1 Counter Register
0x6E2508	LPU Rxlink Unsupported DLLP Received Register
0x6E2510	LPU Rxlink Test Control Register
0x6E2518-0x6E25F8	RESERVED
0x6E2600	LPU Physical Layer Configuration Register
0x6E2608	LPU Phy Layer Status Register
0x6E2610	LPU Phy Layer Interrupt and Status Register
0x6E2618	LPU Phy Interrupt and Status Test Register
0x6E2620	LPU Phy Interrupt Mask Register
0x6E2628-0x6E2678	RESERVED
0x6E2680	LPU Receive Phy Config Register
0x6E2688	LPU Receive Phy Status1 Register
0x6E2690	LPU Receive Phy Status2 Register
0x6E2698	LPU Receive Phy Status3 Register
0x6E26A0	LPU Receive Phy Interrupt and Status Register
0x6E26A8	LPU Receive Phy Interrupt and Status Test Register
0x6E26B0	LPU Receive Phy Interrupt Mask Register
0x6E26B8-0x6E26F8	RESERVED
0x6E2700	LPU Transmit Phy Config Register
0x6E2708	LPU Transmit Phy Status Register
0x6E2710	LPU Transmit Phy Interrupt and Status Register
0x6E2718	LPU Transmit Phy Interrupt and Status Test Register
0x6E2720	LPU Transmit Phy Interrupt Mask Register
0x6E2728	LPU Transmit Phy Status 2 Register

Table 1-19 Fire Register Map (Sheet 11 of 19)

Offset	Register
0x6E2730-0x6E2778	RESERVED
0x6E2780	LPU LTSSM Config1 Register
0x6E2788	LPU LTSSM Config2 Register
0x6E2790	LPU LTSSM Config3 Register
0x6E2798	LPU LTSSM Config4 Register
0x6E27A0	LPU LTSSM Config5 Register
0x6E27A8	LPU LTSSM Status1 Register
0x6E27B0	LPU LTSSM Status2 Register
0x6E27B8	LPU LTSSM Interrupt and Status Register
0x6E27C0	LPU LTSSM Interrupt and Status Test Register
0x6E27C8	LPU LTSSM Interrupt Mask Register
0x6E27D0	LPU LTSSM Status Write Enable Register
0x6E27D8-0x6E27F8	RESERVED
0x6E2800	LPU Serdes Glue Config1 Register
0x6E2808	LPU Serdes Glue Config2 Register
0x6E2810	LPU Serdes Glue Config3 Register
0x6E2818	LPU Serdes Glue Config4 Register
0x6E2820	LPU Serdes Glue Status Register
0x6E2828	LPU Serdes Glue Interrupt and Status Register
0x6E2830	LPU Serdes Glue Interrupt and Status Test Register
0x6E2838	LPU Serdes Glue Interrupt Mask Register
0x6E2840	LPU Serdes Glue Power Down1 Register
0x6E2848	LPU Serdes Glue Power Down2 Register
0x6E2850	LPU Serdes Glue Config5 Register
0x6E2858-0x6FFFF8	RESERVED
PCIE-B CSR Space	
0x700000-0x700FF8	RESERVED
0x701000-0x7011F8	Interrupt Mapping Registers
0x701200-0x7013F8	RESERVED
0x701400-0x7015F8	Interrupt Clear Registers
0x701600-0x7019F8	RESERVED
0x701A00	Interrupt Retry Timer Register
0x701A08	RESERVED
0x701A10	Interrupt State Status Register 1
0x701A18	Interrupt State Status Register 2
0x701A20-0x70AFF8	RESERVED

Table 1-19 Fire Register Map (Sheet 12 of 19)

Offset	Register
0x70B000	INTX Status Register
0x70B008	INT A Clear Register
0x70B010	INT B Clear Register
0x70B018	INT C Clear Register
0x70B020	INT D Clear Register
0x70B028-0x70FFF8	RESERVED
0x710000	Event Queue Base Address Register
0x710008-0x710FF8	RESERVED
0x711000-0x711118	Event Queue Control Set Register
0x711120-0x7111F8	RESERVED
0x711200-0x711318	Event Queue Control Clear Register
0x711320-0x7113F8	RESERVED
0x711400-0x711518	Event Queue State Register
0x711520-0x7115F8	RESERVED
0x711600-0x711718	Event Queue Tail Register
0x711720-0x7117F8	RESERVED
0x711800-0x711918	Event Queue Head Register
0x711920-0x71FFF8	RESERVED
0x720000-0x7207F8	MSI Mapping Register
0x720800-0x727FF8	RESERVED
0x728000-0x7287F8	MSI Clear Registers
0x728800-0x72BFF8	RESERVED
0x72C000	Interrupt Mondo Data 0 Register
0x72C008	Interrupt Mondo Data 1 Register
0x72C010-0x72FFF8	RESERVED
0x730000	ERR COR Mapping Register
0x730008	ERR NONFATAL Mapping Register
0x730010	ERR FATAL Mapping Register
0x730018	PM PME Mapping Register
0x730020	PME To ACK Mapping Register
0x730028-0x730FF8	RESERVED
0x731000	IMU Error Log Enable Register
0x731008	IMU Interrupt Enable Register
0x731010	IMU Interrupt Status Register
0x731018	IMU Error Status Clear Register
0x731020	IMU Error Status Set Register

Table 1-19 Fire Register Map (Sheet 13 of 19)

Offset	Register
0x731028	IMU RDS Error Log Register
0x731030	IMU SCS Error Log Register
0x731038	IMU EQS Error Log Register
0x731040-0x7317F8	RESERVED
0x731800	DMC Core and Block Interrupt Enable Register
0x731808	DMC Core and Block Error Status Register
0x731810	Multi Core Error Status Register
0x731818-0x731FF8	RESERVED
0x732000	IMU Performance Counter Select Register
0x732008	IMU Performance Counter Zero Register
0x732010	IMU Performance Counter One Register
0x732018-0x733FF8	RESERVED
0x734000	MSI 32-bit Address Register
0x734008	MSI 64-bit Address Register
0x734010	RESERVED
0x734018	Mem 64 PCIE Offset Register
0x734020-0x73FFF8	RESERVED
0x740000	MMU Control and Status Register
0x740008	MMU TSB Control Register
0x740010-0x7400F8	RESERVED
0x740100	MMU TTE Cache Flush Address Register
0x740108	MMU TTE Cache Invalidate Register
0x740110-0x740FF8	RESERVED
0x741000	MMU Error Log Enable Register
0x741008	MMU Interrupt Enable Register
0x741010	MMU Interrupt Status Register
0x741018	MMU Error Status Clear Register
0x741020	MMU Error Status Set Register
0x741028	MMU Translation Fault Address Register
0x741030	MMU Translation Fault Status Register
0x741038-0x741FF8	RESERVED
0x742000	MMU Performance Counter Select Register
0x742008	MMU Performance Counter Zero Register
0x742010	MMU Performance Counter One Register
0x742018-0x745FF8	RESERVED
0x746000-0x7461F8	MMU TTE Cache Virtual Tag Registers

Table 1-19 Fire Register Map (Sheet 14 of 19)

Offset	Register
0x746200-0x746FF8	RESERVED
0x747000-0x7471F8	MMU TTE Cache Physical Tag Registers
0x747200-0x747FF8	RESERVED
0x748000-0x748FF8	MMU TTE Cache Data Registers
0x749000-0x750FF8	RESERVED
0x751000	ILU Error Log Enable Register
0x751008	ILU Interrupt Enable Register
0x751010	ILU Interrupt Status Register
0x751018	ILU Error Status Clear Register
0x751020	ILU Error Status Set Register
0x751028-0x7517F8	RESERVED
0x751800	PEC Core and Block Interrupt Enable Register
0x751808	PEC Core and Block Interrupt Status Register
0x751810-0x751FF8	RESERVED
0x752000	ILU Device Capabilities Register
0x752008-0x752FF8	RESERVED
0x753000	DMC Debug Select Register for Port A
0x753008	DMC Debug Select Register for Port B
0x753010-0x7530F8	RESERVED
0x753100	DMC PCI Express Configuration Register
0x753108-0x75FFF8	RESERVED
0x760000-0x7600F8	Packet Scoreboard DMA Register Set
0x760100-0x763FF8	RESERVED
0x764000-0x764078	Packet Scoreboard PIO Register Set
0x764080-0x76FFF8	RESERVED
0x770000-0x7700F8	Transaction Scoreboard Register Set
0x770100	Transaction Scoreboard Status Register
0x770108-0x77FFF8	RESERVED
0x780000	TLU Control Register
0x780008	TLU Status Register
0x780010	TLU PME Turn Off Generate Register
0x780018	TLU Ingress Credits Initial Register
0x780020-0x7800F8	RESERVED
0x780100	TLU Diagnostic Register
0x780108-0x7801F8	RESERVED
0x780200	TLU Egress Credits Consumed Register

Table 1-19 Fire Register Map (Sheet 15 of 19)

Offset	Register
0x780208	TLU Egress Credit Limit Register
0x780210	TLU Egress Retry Buffer Register
0x780218	TLU Ingress Credits Allocated Register
0x780220	TLU Ingress Credits Received Register
0x780228-0x780FF8	RESERVED
0x781000	TLU Other Event Log Enable Register
0x781008	TLU Other Event Interrupt Enable Register
0x781010	TLU Other Event Interrupt Status Register
0x781018	TLU Other Event Status Clear Register
0x781020	TLU Other Event Status Set Register
0x781028	TLU Receive Other Event Header1 Log Register
0x781030	TLU Receive Other Event Header2 Log Register
0x781038	TLU Transmit Other Event Header1 Log Register
0x781040	TLU Transmit Other Event Header2 Log Register
0x781048-0x781FF8	RESERVED
0x782000	TLU Performance Counter Select Register
0x782008	TLU Performance Counter Zero Register
0x782010	TLU Performance Counter One Register
0x782018	TLU Performance Counter Two Register
0x782020-0x782FF8	RESERVED
0x783000	TLU Debug Select A Register
0x783008	TLU Debug Select B Register
0x783010-0x78FFF8	RESERVED
0x790000	TLU Device Capabilities Register
0x790008	TLU Device Control Register
0x790010	TLU Device Status Register
0x790018	TLU Link Capabilities Register
0x790020	TLU Link Control Register
0x790028	TLU Link Status Register
0x790030	TLU Slot Capabilities Register
0x790038-0x790FF8	RESERVED
0x791000	TLU Uncorrectable Error Log Enable Register
0x791008	TLU Uncorrectable Error Interrupt Enable Register
0x791010	TLU Uncorrectable Error Interrupt Status Register
0x791018	TLU Uncorrectable Error Status Clear Register
0x791020	TLU Uncorrectable Error Status Set Register

Table 1-19 Fire Register Map (Sheet 16 of 19)

Offset	Register
0x791028	TLU Receive Uncorrectable Error Header1 Log Register
0x791030	TLU Receive Uncorrectable Error Header2 Log Register
0x791038	TLU Transmit Uncorrectable Error Header1 Log Register
0x791040	TLU Transmit Uncorrectable Error Header2 Log Register
0x791048-0x7A0FF8	RESERVED
0x7A1000	TLU Correctable Error Log Enable Register
0x7A1008	TLU Correctable Error Interrupt Enable Register
0x7A1010	TLU Correctable Error Interrupt Status Register
0x7A1018	TLU Correctable Error Status Clear Register
0x7A1020	TLU Correctable Error Status Set Register
0x7A1028-0x7E1FF8	RESERVED
0x7E2000	LPU ID Register
0x7E2008	LPU Reset Register
0x7E2010	LPU Debug Status Register
0x7E2018	LPU Debug Config Register
0x7E2020	LPU LTSSM Control Register
0x7E2028	LPU Link Status Register
0x7E2030-0x7E2038	RESERVED
0x7E2040	LPU Interrupt Status Register
0x7E2048	LPU Interrupt Mask Register
0x7E2050-0x7E20F8	RESERVED
0x7E2100	LPU Link Performance Counter Select Register
0x7E2108	RESERVED
0x7E2110	LPU Link Performance Counter Control Register
0x7E2118	RESERVED
0x7E2120	LPU Link Performance Counter1
0x7E2128	LPU Link Performance Counter1 Test
0x7E2130	LPU Link Performance Counter2
0x7E2138	LPU Link Performance Counter2 Test
0x7E2140-0x7E21F8	RESERVED
0x7E2200	LPU Link Layer Config Register
0x7E2208	LPU Link Layer Status Register
0x7E2210	LPU Link Layer Interrupt and Status Register
0x7E2218	LPU Link Layer Interrupt and Status Test Register
0x7E2220	LPU Link Layer Interrupt Mask Register
0x7E2228-0x7E2238	RESERVED

Table 1-19 Fire Register Map (Sheet 17 of 19)

Offset	Register
0x7E2240	LPU Flow Control Update Control Register
0x7E2248-0x7E2258	RESERVED
0x7E2260	LPU Link Layer Flow Control Update Timeout Value Register
0x7E2268	LPU Link Layer VC0 Flow Control Update Timer0 Register
0x7E2270	LPU Link Layer VC0 Flow Control Update Timer1 Register
0x7E2278-0x7E23F8	RESERVED
0x7E2400	LPU Txlink Frequent Nak Latency Timer Threshold Register
0x7E2408	LPU Txlink AckNak Latency Timer Register
0x7E2410	LPU Txlink Replay Timer Threshold Register
0x7E2418	LPU Txlink Replay Timer Register
0x7E2420	LPU Txlink Replay Number Status Register
0x7E2428	LPU Replay Buffer Max Address Register
0x7E2430	LPU Txlink Retry FIFO Pointer Register
0x7E2438	LPU Txlink Retry FIFO R/W Pointer Register
0x7E2440	LPU Txlink Retry FIFO Credit Register
0x7E2448	LPU Txlink Sequence Counter Register
0x7E2450	LPU Txlink Ack Sent Sequence Number Register
0x7E2458	LPU Txlink Sequence Count FIFO Max Addr Register
0x7E2460	LPU Txlink Sequence Count FIFO Pointers Register
0x7E2468	LPU Txlink Sequence Count R/W Pointers Register
0x7E2470	LPU Txlink Test Control Register
0x7E2478	RESERVED
0x7E2480	LPU Txlink Memory Address Control Register
0x7E2488	LPU Txlink Memory Data Load0 Register
0x7E2490	LPU Txlink Memory Data Load1 Register
0x7E2498	LPU Txlink Memory Data Load2 Register
0x7E24A0	LPU Txlink Memory Data Load3 Register
0x7E24A8	LPU Txlink Memory Data Load4 Register
0x7E24B0-0x7E24B8	RESERVED
0x7E24C0	LPU Txlink Retry Data Count Register
0x7E24C8	LPU Txlink Sequence Buffer Count Register
0x7E24D0	LPU Txlink Sequence Buffer Bottom Data Register
0x7E24D8	RESERVED
0x7E24E0	LPU Txlink Ack Latency Timer Threshold Register
0x7E24E8-0x7E24F8	RESERVED
0x7E2500	LPU Rxlink Next Receive Sequence + 1 Counter Register

Table 1-19 Fire Register Map (Sheet 18 of 19)

Offset	Register
0x7E2508	LPU Rxlink Unsupported DLLP Received Register
0x7E2510	LPU Rxlink Test Control Register
0x7E2518-0x7E25F8	RESERVED
0x7E2600	LPU Physical Layer Configuration Register
0x7E2608	LPU Phy Layer Status Register
0x7E2610	LPU Phy Layer Interrupt and Status Register
0x7E2618	LPU Phy Interrupt and Status Test Register
0x7E2620	LPU Phy Interrupt Mask Register
0x7E2628-0x7E2678	RESERVED
0x7E2680	LPU Receive Phy Config Register
0x7E2688	LPU Receive Phy Status1 Register
0x7E2690	LPU Receive Phy Status2 Register
0x7E2698	LPU Receive Phy Status3 Register
0x7E26A0	LPU Receive Phy Interrupt and Status Register
0x7E26A8	LPU Receive Phy Interrupt and Status Test Register
0x7E26B0	LPU Receive Phy Interrupt Mask Register
0x7E26B8-0x7E26F8	RESERVED
0x7E2700	LPU Transmit Phy Config Register
0x7E2708	LPU Transmit Phy Status Register
0x7E2710	LPU Transmit Phy Interrupt and Status Register
0x7E2718	LPU Transmit Phy Interrupt and Status Test Register
0x7E2720	LPU Transmit Phy Interrupt Mask Register
0x7E2728	LPU Transmit Phy Status 2 Register
0x7E2730-0x7E2778	RESERVED
0x7E2780	LPU LTSSM Config1 Register
0x7E2788	LPU LTSSM Config2 Register
0x7E2790	LPU LTSSM Config3 Register
0x7E2798	LPU LTSSM Config4 Register
0x7E27A0	LPU LTSSM Config5 Register
0x7E27A8	LPU LTSSM Status1 Register
0x7E27B0	LPU LTSSM Status2 Register
0x7E27B8	LPU LTSSM Interrupt and Status Register
0x7E27C0	LPU LTSSM Interrupt and Status Test Register
0x7E27C8	LPU LTSSM Interrupt Mask Register
0x7E27D0	LPU LTSSM Status Write Enable Register
0x7E27D8-0x7E27F8	RESERVED

Table 1-19 Fire Register Map (Sheet 19 of 19)

Offset	Register
0x7E2800	LPU Serdes Glue Config1 Register
0x7E2808	LPU Serdes Glue Config2 Register
0x7E2810	LPU Serdes Glue Config3 Register
0x7E2818	LPU Serdes Glue Config4 Register
0x7E2820	LPU Serdes Glue Status Register
0x7E2828	LPU Serdes Glue Interrupt and Status Register
0x7E2830	LPU Serdes Glue Interrupt and Status Test Register
0x7E2838	LPU Serdes Glue Interrupt Mask Register
0x7E2840	LPU Serdes Glue Power Down1 Register
0x7E2848	LPU Serdes Glue Power Down2 Register
0x7E2850	LPU Serdes Glue Config5 Register

1.3.3 JBus Registers

1.3.3.1 JBus Device ID Register (0x00000000 / 0xFC0000000390000)

This is a SW read only register which is the only register living in the Fcode region and the only address supported in that region. It contains id information about the device including valid JBUS ports, current JBUS ID, and manufacturing ID's.

Table 1-20 JBus Device ID Register

Field	Bits	Reset Name	Reset Value	Type	Description
COOKIE	63:56	rst_1	0xFC	R	This field is provided so that the open boot PROM code detects that Fire does not support an Fcode PROM.
RESERVED	55:34				Reserved field
JVPORT	33:27	rst_1	7'bx	R	Valid JBus ports in current system configuration. Each bit of this field is used to reflect the presence of a JBus Port in the system. If set to one, the JBus Port is invalid and if zero, the JBus Port is valid. The value of J_PACK[2:0] for each port is sampled after reset deassertion, if DTL pull-up on the J_PACK pins indicate pulled up state, the corresponding port is considered to be invalid for the current configuration. This is the same as Tomatillo.
RESERVED	26:22				Reserved field
JPID_4	21	rst_1	0x1	R	Most significant bit of Fire's JBus Port ID. This is hardcoded to a value of 1. It is NOT Programmable

Table 1-20 JBus Device ID Register

Field	Bits	Reset Name	Reset Value	Type	Description
JPID_3_0	20:17	rst_1	4'b11x0	R	Least Significant 4 bits of Fire's JBus Port ID. Since Fire can only be in two slots in a JBUS system and this value must report the first JBUS PortID that Fire occupies, the id's are limited This value will be as follows: {1,1,j_id[1],0} This value of j_id[1] will be set after reset from the external J_ID[1] pin on Fire.
M_S	16	rst_1	0x1	R	Master/Slave. Fire is both a master and slave device on JBus. This Register is set to a value of 1 for all Fire's in a system and can never be changed
MID	15:10	rst_1	6'bx	R	Manufacturer ID: 0x36 read only in device id, same as jtagid bits 11:1, connects to bits 6:1 of jtagid ignore the reset value
MT	9:4	rst_1	6'bx	R	Module Type: 0x22 read only in device id, same as jtagid bits 27:12, connects to bits 17:12 of jtagid ignore the reset value
MR	3:0	rst_1	4'bx	R	Module Revision: 0x4 for Fire 2.1 read only in device id, same as jtagid bits 31:28, connects to bits 31:28 of jtagid ignore the reset value

1.3.3.2 EBus Offset Base Register (0x00400020 / 0x8000000FF0000000)

This is the EBus Offset Base Register which is used to enable the EBus portion of the 64 G non-cacheable region for use. It contains a valid bit and the actual address offset used to match a JBUS transaction against to determine if that transaction is to the EBus region. This register defaults to having this region enabled.

Table 1-21 EBus Offset Base Register

Field	Bits	Reset Name	Reset Value	Type	Description
V	63	rst_1	0x1	RW	Valid bit to determine if the address region enabled. If set to 1 address region is enabled. If set to 0 address region is not enabled. If not enabled writes are silently dropped, reads return unmapped error status.
RESERVED	62:36				Reserved field
BASE	35:24	rst_1	0xFF0	RW	Base Value. This is the value of the base address to be matched against. Depending on the Mask Bits, software should load Base bits corresponding to Mask bits = 0 with all zeroes.
RESERVED	23:0				Reserved field

1.3.3.3 EBus Offset Mask Register (0x00400028 / 0x7FFF80000000)

This is the EBus Offset Mask Register which is used mask out certain bits of the address for a given JBUS transaction before it is used to match with the offset base register of the EBus region.

Table 1-22 EBus Offset Mask Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:43				Reserved field
MASK_HI	42:36	rst_1	0x7F	R	High order mask bits, always set to 1s.

Table 1-22 EBus Offset Mask Register

Field	Bits	Reset Name	Reset Value	Type	Description
MASK	35:24	rst_l	0xFF8	RW	Mask Value. This mask is used to remove the least significant bits before matching. It defines the size of the mapped region. It must be string of 1 followed by a string of 0.
RESERVED	23:0				Reserved field

1.3.3.4 PCIE-A Mem32 Offset Base Register (0x00400040 / 0x0)

This is the PCIE-A Mem32 Offset Base Register which is used to enable the PCIE-A Mem32 portion of the 64 G non-cacheable region for use. It contains a valid bit and the actual address offset used to match a JBUS transaction against to determine if that transaction is to the PCIE-A Mem32 region. This register defaults to having this region enabled.

Table 1-23 PCIE-A Mem32 Offset Base Register

Field	Bits	Reset Name	Reset Value	Type	Description
V	63	rst_l	0x0	RW	Valid bit to determine if the address region enabled. If set to 1 address region is enabled. If set to 0 address region is not enabled. If not enabled writes are silently dropped, reads return unmapped error status.
RESERVED	62:36				Reserved field
BASE	35:24	rst_l	0x0	RW	Base Value. This is the value of the base address to be matched against. Depending on the Mask Bits, software should load Base bits corresponding to Mask bits = 0 with all zeroes.
RESERVED	23:0				Reserved field

1.3.3.5 PCIE-A Mem32 Offset Mask Register (0x00400048 / 0x7F00000000)

This is the PCIE-A Mem32 Offset Mask Register which is used mask out certain bits of the address for a given JBUS transaction before it is used to match with the offset base register of the PCIE-A Mem32 region.

Table 1-24 PCIE-A Mem32 Offset Mask Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:43				Reserved field
MASK_HI	42:36	rst_1	0x7F	R	High order mask bits, always set to 1s.
MASK	35:24	rst_1	0x0	RW	Mask Value. This mask is used to remove the least significant bits before matching. It defines the size of the mapped region. It must be string of 1 followed by a string of 0.
RESERVED	23:0				Reserved field

1.3.3.6 PCIE-A Cfg/IO Offset Base Register (0x00400050 / 0x0)

This is the PCIE-A Cfg/IO Offset Base Register which is used to enable the PCIE-A Cfg/IO portion of the 64 G non-cacheable region for use. It contains a valid bit and the actual address offset used to match a JBUS transaction against to determine if that transaction is to the PCIE-A Cfg/IO region. This register defaults to having this region enabled.

Table 1-25 PCIE-A Cfg/IO Offset Base Register

Field	Bits	Reset Name	Reset Value	Type	Description
V	63	rst_1	0x0	RW	Valid bit to determine if the address region enabled. If set to 1 address region is enabled. If set to 0 address region is not enabled. If not enabled writes are silently dropped, reads return unmapped error status.
RESERVED	62:36				Reserved field

Table 1-25 PCIE-A Cfg/IO Offset Base Register

Field	Bits	Reset Name	Reset Value	Type	Description
BASE	35:24	rst_l	0x0	RW	Base Value. This is the value of the base address to be matched against. Depending on the Mask Bits, software should load Base bits corresponding to Mask bits = 0 with all zeroes.
RESERVED	23:0				Reserved field

1.3.3.7 PCIE-A Cfg/IO Offset Mask Register (0x00400058 / 0x7F00000000)

This is the PCIE-A Cfg/IO Offset Mask Register which is used mask out certain bits of the address for a given JBUS transaction before it is used to match with the offset base register of the PCIE-A Cfg/IO region.

Table 1-26 PCIE-A Cfg/IO Offset Mask Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:43				Reserved field
MASK_HI	42:36	rst_l	0x7F	R	High order mask bits, always set to 1s.
MASK	35:24	rst_l	0x0	RW	Mask Value. This mask is used to remove the least significant bits before matching. It defines the size of the mapped region. It must be string of 1 followed by a string of 0.
RESERVED	23:0				Reserved field

1.3.3.8 PCIE-B Mem32 Offset Base Register (0x00400060/0x0)

This is the PCIE-B Mem32 Offset Base Register which is used to enable the PCIE-B Mem32 portion of the 64 G non-cacheable region for use. It contains a valid bit and the actual address offset used to match a JBUS transaction against to determine if that transaction is to the PCIE-B Mem32 region. This register defaults to having this region enabled.

Table 1-27 PCIE-B Mem32 Offset Base Register

Field	Bits	Reset Name	Reset Value	Type	Description
V	63	rst_l	0x0	RW	Valid bit to determine if the address region enabled. If set to 1 address region is enabled. If set to 0 address region is not enabled. If not enabled writes are silently dropped, reads return unmapped error status.
RESERVED	62:36				Reserved field
BASE	35:24	rst_l	0x0	RW	Base Value. This is the value of the base address to be matched against. Depending on the Mask Bits, software should load Base bits corresponding to Mask bits = 0 with all zeroes.
RESERVED	23:0				Reserved field

1.3.3.9 PCIE-B Mem32 Offset Mask Register (0x00400068/0x7F00000000)

This is the PCIE-B Mem32 Offset Mask Register which is used mask out certain bits of the address for a given JBUS transaction before it is used to match with the offset base register of the PCIE-B Mem32 region.

Table 1-28 PCIE-B Mem32 Offset Mask Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:43				Reserved field
MASK_HI	42:36	rst_l	0x7F	R	High order mask bits, always set to 1s.

Table 1-28 PCIE-B Mem32 Offset Mask Register

Field	Bits	Reset Name	Reset Value	Type	Description
MASK	35:24	rst_1	0x0	RW	Mask Value. This mask is used to remove the least significant bits before matching. It defines the size of the mapped region. It must be string of 1 followed by a string of 0.
RESERVED	23:0				Reserved field

1.3.3.10 PCIE-B Cfg/IO Offset Base Register (0x00400070 / 0x0)

This is the PCIE-B Cfg/IO Offset Base Register which is used to enable the PCIE-B Cfg/IO portion of the 64 G non-cacheable region for use. It contains a valid bit and the actual address offset used to match a JBUS transaction against to determine if that transaction is to the PCIE-B Cfg/IO region. This register defaults to having this region enabled.

Table 1-29 PCIE-B Cfg/IO Offset Base Register

Field	Bits	Reset Name	Reset Value	Type	Description
V	63	rst_1	0x0	RW	Valid bit to determine if the address region enabled. If set to 1 address region is enabled. If set to 0 address region is not enabled. If not enabled writes are silently dropped, reads return unmapped error status.
RESERVED	62:36				Reserved field
BASE	35:24	rst_1	0x0	RW	Base Value. This is the value of the base address to be matched against. Depending on the Mask Bits, software should load Base bits corresponding to Mask bits = 0 with all zeroes.
RESERVED	23:0				Reserved field

1.3.3.11 PCIE-B Cfg/IO Offset Mask Register (0x00400078 / 0x7F00000000)

This is the PCIE-B Cfg/IO Offset Mask Register which is used mask out certain bits of the address for a given JBUS transaction before it is used to match with the offset base register of the PCIE-B Cfg/IO region.

Table 1-30 PCIE-B Cfg/IO Offset Mask Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:43				Reserved field
MASK_HI	42:36	rst_1	0x7F	R	High order mask bits, always set to 1s.
MASK	35:24	rst_1	0x0	RW	Mask Value. This mask is used to remove the least significant bits before matching. It defines the size of the mapped region. It must be string of 1 followed by a string of 0.
RESERVED	23:0				Reserved field

1.3.3.12 PCIE-A Mem64 Offset Base Register (0x00400080 / 0x0)

This is the PCIE-A Mem64 Offset Base Register which is used to enable the PCIE-A Mem64 portion of the 64 G non-cacheable region for use. It contains a valid bit and the actual address offset used to match a JBUS transaction against to determine if that transaction is to the PCIE-A Mem64 region. This register defaults to having this region enabled.

Table 1-31 PCIE-A Mem64 Offset Base Register

Field	Bits	Reset Name	Reset Value	Type	Description
V	63	rst_1	0x0	RW	Valid bit to determine if the address region enabled. If set to 1 address region is enabled. If set to 0 address region is not enabled. If not enabled writes are silently dropped, reads return unmapped error status.
RESERVED	62:36				Reserved field

Table 1-31 PCIE-A Mem64 Offset Base Register

Field	Bits	Reset Name	Reset Value	Type	Description
BASE	35:24	rst_1	0x0	RW	Base Value. This is the value of the base address to be matched against. Depending on the Mask Bits, software should load Base bits corresponding to Mask bits = 0 with all zeroes. The PCI-Express Mem64 PIO address is formed from {offset[63:36], (offset[35:24] jbus addr[35:24]), jbus addr[23:2]} where jbus addr[35:0] = {(PA[35:24] & ~mask[35:24]), PA[23:0]} and offset[63:24] is from the Mem64 PCIE Offset Register
RESERVED	23:0				Reserved field

1.3.3.13 PCIE-A Mem64 Offset Mask Register (0x00400088 / 0x7F00000000)

This is the PCIE-A Mem64 Offset Mask Register which is used mask out certain bits of the address for a given JBUS transaction before it is used to match with the offset base register of the PCIE-A Mem64 region.

Table 1-32 PCIE-A Mem64 Offset Mask Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:43				Reserved field
MASK_HI	42:36	rst_1	0x7F	R	High order mask bits, always set to 1s.
MASK	35:24	rst_1	0x0	RW	Mask Value. This mask is used to remove the least significant bits before matching. It defines the size of the mapped region. It must be string of 1 followed by a string of 0.
RESERVED	23:0				Reserved field

1.3.3.14 PCIE-B Mem64 Offset Base Register (0x00400090/0x0)

This is the PCIE-B Mem64 Offset Base Register which is used to enable the PCIE-B Mem64 portion of the 64 G non-cacheable region for use. It contains a valid bit and the actual address offset used to match a JBUS transaction against to determine if that transaction is to the PCIE-B Mem64 region. This register defaults to having this region enabled.

Table 1-33 PCIE-B Mem64 Offset Base Register

Field	Bits	Reset Name	Reset Value	Type	Description
V	63	rst_1	0x0	RW	Valid bit to determine if the address region enabled. If set to 1 address region is enabled. If set to 0 address region is not enabled. If not enabled writes are silently dropped, reads return unmapped error status.
RESERVED	62:36				Reserved field
BASE	35:24	rst_1	0x0	RW	Base Value. This is the value of the base address to be matched against. Depending on the Mask Bits, software should load Base bits corresponding to Mask bits = 0 with all zeroes. The PCI-Express Mem64 PIO address is formed from {offset[63:36], (offset[35:24] jbus addr[35:24]), jbus addr[23:2]} where jbus addr[35:0] = {(PA[35:24] & ~mask[35:24]), PA[23:0]} and offset[63:24] is from the Mem64 PCIE Offset Register
RESERVED	23:0				Reserved field

1.3.3.15 PCIE-B Mem64 Offset Mask Register (0x00400098 / 0x7F00000000)

This is the PCIE-B Mem64 Offset Mask Register which is used mask out certain bits of the address for a given JBUS transaction before it is used to match with the offset base register of the PCIE-B Mem64 region.

Table 1-34 PCIE-B Mem64 Offset Mask Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:43				Reserved field
MASK_HI	42:36	rst_1	0x7F	R	High order mask bits, always set to 1s.
MASK	35:24	rst_1	0x0	RW	Mask Value. This mask is used to remove the least significant bits before matching. It defines the size of the mapped region. It must be string of 1 followed by a string of 0.
RESERVED	23:0				Reserved field

1.3.3.16 Fire Control/Status Register (0x00410000 / 0x7F0038B6000)

This is the Fire Control/Status Register, it allows SW to control and get status information about certain major modes and functions of Fire. This register allows SW to control Fire's parity modes, DTL modes, arbitrations modes, error reporting modes, flow control thresh holds, and timeout settings. This register in addition to the above also allows software to obtain Fires Port ID and whether it is in Niagara mode or not.

The JBus interface has a free-running watchdog timer which can be enabled into any one of the 'enabled' states through software, or may be disabled. This timer is used to detect timeouts on reads, writes and interrupts. The timer counts continuously, wrapping when it reaches the value programmed in bits [33:32] of this register. Time information is stored for each of the 32 transactions in the DMCINT scoreboard using 2 bits each. When the timer wraps a bit is set. When both bits have set for a current transaction still waiting for it's completion information, a timeout for that transaction is flagged. Hence a timeout will occur within the window of 1X to 2X the timer's programmed value. The wrap bits are cleared when a new transaction is written into the scoreboard. A transaction is considered complete when either the datum for a read or the ack/nack for an interrupt is returned to Fire over JBus or when the write ack is received from the JBUSINT block, which occurs when the write is issued on the JBus. Any of the timeouts will cause a fatal error and a reset on the JBus. Timeouts may be changed between 'disabled' and any 'enabled' state without causing spurious timeouts. When in the 'disabled' state, timeouts are frozen and the interval is literally infinite.

Caution: Changing the timeout interval between 'enabled' states may cause spurious timeouts. Changing the timeout interval from one 'enabled' state, to the 'disabled' state, and then to a different 'enabled' state may timeout in the shorter of the two intervals.

Table 1-35 Fire Control/Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
SPARE_CONTROL_LOAD_4	63	rst_1	0x0	RW	Spare Loadable Control Register 4
SPARE_CONTROL_LOAD_3	62	rst_1	0x0	RW	Spare Loadable Control Register 3
SPARE_CONTROL_LOAD_2	61	rst_1	0x0	RW	Spare Loadable Control Register 2
SPARE_CONTROL_LOAD_1	60	rst_1	0x0	RW	Spare Loadable Control Register 1
SPARE_CONTROL_LOAD_0	59	rst_1	0x0	RW	Spare Loadable Control Register 0
SPARE_CONTROL	58:54	rst_1	0x0	RW	Spare Control Registers
SPARE_STATUS	53:49	rst_1	0x0	RW	Spare Status Registers
RESERVED	48:45				Reserved field
PAR_DELAY	44	rst_1	0x0	RW	Delay internally generated control parity by one clock if set to 1. This is needed to align the internally generated control parity with the J_PAR on the pin which can come through repeaters in the system with multiple JBus segments, before comparing against J_PAR.
PAR_EN	43	rst_1	0x0	RW	Control Parity Error Check Enable. The bit encodings are: 0: Control Parity Error Check Disabled. 1: Control Parity Error Check Enabled.

Table 1-35 Fire Control/Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
JPACK_DELAY	42:36	rst_l	0x7F	RW	<p>J_pack delay control bit for each port. Bit 0 corresponds to J_PACK0, bit 6 to J_PACK6. the control bit encodings are:</p> <p>0: Do not delay J_PACK for that port into the parity check generate logic. Bit should be 0 if JBus port is on a remote JBus segment.</p> <p>1: Delay J_PACK by one clock for that port into the parity check/generate logic. Bit should be 1 if JBus port is on a local JBus segment.</p> <p>Note: Reset state is 0x7f. So the implication is at reset, all JBus ports should have the control parity check disabled. Fire has its own parity check disabled at reset. Have to make sure that the CPUs also do the same thing. Once software comes and updates this register in all JBus ports, then all the ports should have their control parity error detection enable turned on, other than Fire with JPID[3:1] == 7, which should always drive J_PAR and not detect control parity error. This is the same as Tomatillo.</p>

Table 1-35 Fire Control/Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
DTL_MODE	35:34	rst_1	2'bx	R	DTL mode for Driver/Receiver group. The Mode Encodings are: 0x0 - Reserved 0x1 - DTL-end mode 0x2 - DTL-mid mode 0x3 - DTL-2 mode This field is read-only here. The DTL modes are set upon reset from 2 external hardwired pins in Fire as per the following relationship: $dtl_mode = \{j_down25, \sim j_upopen\}$, where: j_down25 = 25 ohm pull down enable j_upopen = 50 ohm pull-up disable
JTO	33:32	rst_1	0x0	RW	JBus watchdog timer timeout interval. Values are: 0x0 => oo (JBus timeouts disabled) 0x1 => 2^{28} JBus clock cycles (normal) 0x2 => 2^{15} JBus clock cycles (debug) 0x3 => 2^9 JBus clock cycles (simulation) This controls the duration of each JBus time-out event defined in JBus Error Status Clear Register See cautions on changing JTO while any time-out events are enabled.
RESERVED	31:29				Reserved field

Table 1-35 Fire Control/Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
ARB_MODE	28:27	por_1	0x0	RW	<p>Arbitration Mode for DEAD cycle in between switching drivers. The valid encodings are:</p> <p>00: One dead cycle between driver switch. All JBus ports sample J_ADTYPE, J_AD, and J_ADP during dead cycle. This is the only mode for the system with multiple JBus segments. During dead cycle, termination logic pulls up J_AD, J_ADTYPE and J_ADP to all 1 s.</p> <p>01: One dead cycle between driver switch. No JBus port samples J_ADTYPE, J_AD, and J_ADP during dead cycle. This mode is used only for the highest frequency single-bus systems, where the bus is operating at such a high frequency that dead cycles do not meet timing. During dead cycle, JBus is HighZ.</p> <p>10: No dead cycle between driver switches. All JBus ports sample J_ADTYPE, J_AD, and J_ADP during all cycles.</p> <p>11: Reserved.</p>

Table 1-35 Fire Control/Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
UE_PROP_MODE	26	rst_1	0x0	RW	<p>The mode bit controls how Fire responds to bad RDO data received during processing of a partial write transaction.</p> <p>OFF (1'b0) - If one or more UE errors occur on RDO (RDR64) data during a partial write, then the partial write data is thrown away, and the RDO data is written back to memory, unmerged, without a UE error specified on any of the four data beats of the ensuing WRB. If a 16 byte read data error packet is returned for the RDO instead of a normal RDR64, then the partial write data will be written back to memory, unmerged, with a UE error indicated on all four write data cycles of the WRB.</p> <p>ON (1'b1) - If one or more UE errors occur on RDO (RDR64) data during a partial write, then the RDR64 data and partial write data are merged as usual, but each data cycle of the RDR64 containing an error results in a UE error being indicated during the corresponding write data cycle of the WRB. If a 16 byte read data error packet is returned for the RDO instead of a normal RDR64, then the partial write data will be written back to memory, unmerged, with a UE error indicated on all four write data cycles of the WRB.</p>
JPID_4	25	rst_1	0x1	R	<p>Most significant bit of Fire's JBus Port ID. This is hardcoded to a value of 1. It IS NOT Programmable</p>

Table 1-35 Fire Control/Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
JPID_3_0	24:21	rst_1	4'b11x0	R	Least Significant 4 bits of Fire's JBus Port ID. Since Fire can only be in two slots in a JBUS system and this value must report the first JBUS PortID that Fire occupies, the id's are limited This value will be as follows: {1,1,j_id[1],0} This value of j_id[1] will be set after reset from the external J_ID[1] pin on Fire.
AOK_THRESH	20:17	rst_1	0x5	RW	AOK threshold for Snoop queue. Warning - while this is a program-mable value, changing it should only be done as a debug lab fix and the value MUST be set in the range of 3 - 5 in repeater mode or 3 - 7 in non-repeater mode. Fire will drive AOK OFF when the snoop queue depth is equal to AOK_THRESH - 2,
DOK_THRESH	16:13	rst_1	0xB	RW	DOK threshold for PIO WR Data queue. Warning - while this is a program-mable value, changing it should only be done as a debug lab fix and the value MUST be set in the range of 1 - 11. Fire will drive DOK OFF when PIO WR data queue depth is equal to DOK_THRESH.
NIAGARA_MODE	12	por_1	1'bx	R	Niagara Mode, used to determine if Fire is in Niagara mode or not. It's directly connected to an out-side pin. Ignore the reset value, driven from pin on chip

Table 1-35 Fire Control/Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
PDQ	11:10	rst_1	0x0	RW	<p>Programmable DTL Output Delay. Specifies driver (EN -> IO flop) clock to IO delay time.</p> <p>0x0 - min 400 ps max 900 ps 0x1 - min 600 ps max 1000 ps 0x2 - min 700 ps max 1100 ps 0x3 - min 800 ps max 1200 ps</p> <p>Note that while in Estar 1/2 or 1/32 speed mode, these bits will be forced to 0x3 in hardware.</p>
J_AD4_DIAG	9	rst_1	0x0	RW	<p>Select line for driving diagnostic information on J_AD[4].</p> <p>If 0, J_AD[4] will be 0.</p> <p>If 1, J_AD[4] will be driven by Fire with diagnostic information specifying which PCI-E core the transaction originated from. 0 = PCI-E A, 1 = PCI-E B.</p>
LPDQ	8:0	rst_1	0x0	RW	<p>Large Programmable DTL Output Delay. Specifies driver (EN -> IO flop) clock to IO when PDQ bits are set to 0x3. Used for tester loop-back and Estar hold time modes.</p> <p>The LPDQ value is JTAG Shadow RW.</p>

1.3.3.17 JBus PLL Control and DTL Control Register (0x00410050 / 0x6)

Due to some timing limitations in Fire 2.0, the en and ep signals from the DTL Controller which update on a regular basis do not meet timing and are multi cycle paths. For this reason, the register will have to be read multiple times to determine if the value is stable and not in transition due to the timing violations. Software should

read this register at least 2 times in a row and only use the value as stable if the data for the two consecutive reads matches. This effects the en and ep bits in register bits 59:40

Table 1-36 JBus PLL Control and DTL Override Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:60				Reserved field
EP_50	59:55	por_1	0x0	R	Used to read the status the DTL ctrl codes
EP_25	54:50	por_1	0x0	R	Used to read the status the DTL ctrl codes
EN_50	49:45	por_1	0x0	R	Used to read the status the DTL ctrl codes
EN_25	44:40	por_1	0x0	R	Used to read the status the DTL ctrl codes
DTL_TST2_SCHEME	39	por_1	0x0	RW	Selects between DTL Scheme-1 and Scheme-2 for the DTL_TST2 tight I/O characterization pins. 0 -> Scheme 1, 1 -> Scheme 2
EP_50_O	38:32	por_1	0x0	RW	Used to override the DTL ctrl codes [6] - bias/force; 0 => bias, 1 => force [5] - increment/decrement; 0 => inc, 1 => dec [4:0] - bias code [2:0], or force code [4:0] a bias code of 3'b000 means no change to the generated code
RESERVED	31				Reserved field
EP_25_O	30:24	por_1	0x0	RW	Used to override the DTL ctrl codes [6] - bias/force; 0 => bias, 1 => force [5] - increment/decrement; 0 => inc, 1 => dec [4:0] - bias code [2:0], or force code [4:0] a bias code of 3'b000 means no change to the generated code

Table 1-36 JBus PLL Control and DTL Override Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	23				Reserved field
EN_50_O	22:16	por_1	0x0	RW	Used to override the DTL ctrl codes [6] - bias/force; 0 => bias, 1 => force [5] - increment/decrement; 0 => inc, 1 => dec [4:0] - bias code [2:0], or force code [4:0] a bias code of 3'b000 means no change to the generated code
DTL_FAST_UPDATE	15	por_1	0x0	RW	Force fast DTL controller updates for characterization: 0 - Standard update interval 1 - Fast update interval
EN_25_O	14:8	por_1	0x0	RW	Used to override the DTL ctrl codes [6] - bias/force; 0 => bias, 1 => force [5] - increment/decrement; 0 => inc, 1 => dec [4:0] - bias code [2:0], or force code [4:0] a bias code of 3'b000 means no change to the generated code
DTL_TST2_MODE	7:6	por_1	0x0	RW	Used to select the type of test that the J_DTL_TST2 pins are used for: 00 - 3-node loopback test, default 01 - Setup test, turbo and standard cells 10 - QI test, turbo cell 11 - QI test, standard cell
PLL_LOCK	5	por_1	0x0	R	CSR Status Bit which tells if the PLL is locked. Copy of g2j_pll_lock 1 cycle delayed

Table 1-36 JBus PLL Control and DTL Override Register

Field	Bits	Reset Name	Reset Value	Type	Description
DTL_CHAR	4	por_1	0x0	RW	<p>FOR TESTER ONLY! Used to configure normally endpoint-only DTL I/Os to match Fire's J33_UPOPEN, J33_DOWN25 mode during characterization. These include J_REQ_OUT_L*, J_REQ_IN_L*, and J_ERR. This bit should never be set in a functional system.</p> <p>0x0 - Normal Fire DTL configuration</p> <p>0x1 - All Fire DTL I/Os controlled via J33_UPOPEN/J33_DOWN25 for characterization</p>
JITLMT	3:2	por_1	0x1	R	<p>Jitter threshold range (PLL will assert Fire IO F_PLL_TST and signal g2j_pll_lock to indicate that lock has occurred within min/max jitter range):</p> <p>0x0=>min:94ps, max:240ps</p> <p>0x1=>min:180ps, max:490ps</p> <p>0x2=>min:410ps, max:1100ps</p> <p>0x3=>min:820ps, max:2000ps</p>
CNTLMT	1:0	por_1	0x2	R	<p>Jitter threshold count register (Number of consecutive cycles PLL was in lock.).</p> <p>0x0=>128</p> <p>0x1=>256</p> <p>0x2=>512</p> <p>0x3=>1024</p>

1.3.3.18 JBus Energy Star Control Register (0x00410058/0x1)

One and only one bit shall be set at a time in this register.

Table 1-37 JBus Energy Star Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:6				Reserved field
S1_32	5	por_1	0x0	RW	Operate JBus and interface logic at 1/32nd input frequency. Power Up reset = 0. Preserves its value across soft reset.
RESERVED	4:2				Reserved field
S1_2	1	por_1	0x0	RW	Operate JBus and interface logic at 1/2 the frequency of the input clock. Power Up reset = 0. Preserves its value across soft reset.
FULL	0	por_1	0x1	RW	Operate JBus and interface logic at the same frequency as the input clock. Power Up reset = 1. Preserves its value across soft reset.

1.3.3.19 JBus Change Initiation Control Register (0x00410060/0x0)

Table 1-38 JBus Change Initiation Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:5				Reserved field
CINIT	4:3	por_1	0x0	RW	Change Init. Software writes 2'b10 to initiate the Change sequence. A write of 2'b11 is a software error. Fire then initiates Change transaction on JBus and changes this field to 2'b11. Software then comes and reads 2'b11 and clears it to 2'b00. Power Up reset = 0x0. Preserves its value across soft reset.

Table 1-38 JBus Change Initiation Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
CDELAY	2:0	por_1	0x0	RW	Change Delay. J_CHANGE_L propagation cycles in the system to all valid JBus agents. Valid values are 0, 1, 2, 3, 4, 5. Set by OBP. This represents the delay in JBus clock cycles in using the J_CHNG_L internally to match the delay that all other valid JBus agents see in getting J_CHNG_L. The recommended Programming Value is 0x0 for a system without a repeater and 0x2 for a system with a repeater. Power Up reset = 0x0. Preserves its value across soft reset.

1.3.3.20 Reset Generation Register (0x00417010/0x0)

This is the Reset Generation Register. This register allows SW to initiate one of three types of reset; a hard reset, a soft reset, or an XIR. Only one bit should be set in this register at once.

If Fire is the boot device, this register allows reset to all processors and Fire to facilitate a system reset. The bits in the Reset_Gen registers are not sticky, thus preventing a continuous reset from occurring. If Fire is a slave device, writes to this register have no effect on Fire or the system.

Table 1-39 Reset Generation Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:3				Reserved field
PU_RST	2	rst_1	0x0	RW	Software Hard Reset. Resets all processors when set to 1 by asserting J_POR_L and J_RST_L. Set by software, cleared by soft reset.
XIR	1	rst_1	0x0	RW	Software XIR. Generate XIR transaction when set to 1. Neither J_POR_L nor J_RST_L are asserted. Set by software, cleared by hardware.

Table 1-39 Reset Generation Register

Field	Bits	Reset Name	Reset Value	Type	Description
PO_RST	0	rst_l	0x0	RW	Software Soft Reset. Resets all processors when set to 1 by asserting J_RST_L. Set by software, cleared by soft rst.

1.3.3.21 Reset Source Register (0x00417018/0x0)

The origin of a reset is identified by reading the contents of Reset_Source Register. When a system reset/XIR occurs a bit is set in this register so that the cause of the reset can be determined. This register must be cleared after being read to avoid ambiguity on subsequent resets. Bits are cleared by writing them to one (write one to clear).

The Reset_Source Register logs the cause of any local or external POR taken by a processor so that the POR trap handler can easily identify the source. Either of the low order three bits is set to one when the corresponding bit is set in the Reset_Gen register.

Table 1-40 Reset Source Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:7				Reserved field
FATAL	6	por_l	0x0	RW1C	Fatal Error. Fire causes a Soft reset of the system by asserting J_RST_L. Fatal error gets logged due to detection of the following errors in Fire: mb_pea, cpe, ape, jtceei, jtceer, jtceew, pio_cpe or when Fire detects a fatal error from another JBus port by sampling 3 b111 on j_pack inputs for 4 consecutive clocks. Preserves its value across Soft reset.
PB_XIR	5	por_l	0x0	RW1C	Button XIR/WatchDog Timer Expired. Processors receive XIR due to Button XIR/Watchdog Timer Expiration. Preserves its value across soft reset.

Table 1-40 Reset Source Register

Field	Bits	Reset Name	Reset Value	Type	Description
PB_RST	4	por_l	0x0	RW1C	Push-Button Reset (Power_Good stays high while PB_RST_L asserted). Jalapenos soft reset through pushbutton in the system. J_RST_L asserted to the Jalapenos. Preserves its value across soft reset.
PU	3	por_l	0x0	RW1C	Power_up (Low to High transition on Power_Good). System reset by being powered-up. J_POR_L and J_RST_L asserted to the Jalapenos. Preserves its value across soft reset.
PU_RST	2	por_l	0x0	RW1C	Software Hard Reset. Processors Reset through J_POR_L and J_RST_L pins due to setting of Soft_Pwr_Up_Rst bit of Reset_Gen register. Preserves its value across soft reset.
XIR	1	por_l	0x0	RW1C	Software XIR. Processors received XIR through Soft_XIR bit of Reset_Gen register. Preserves its value across soft reset.
PO_RST	0	por_l	0x0	RW1C	Software Soft Reset. Processors Reset through J_RST_L pin due to setting of Soft_Pwr_On_Rst bit of Reset_Gen register. Preserves its value across soft reset.

1.3.3.22 GPIO Port 0 Pin 0 Data Register (0x00460000/0x0)

This is GPIO Port 0 Pin 0 Data Register. This register allows SW to read the value of pin 0 on GPIO port 0 when it is configured as an input and allows SW to write a value to pin 0 on GPIO port 0 when it is configured as an output.

Table 1-41 GPIO Port 0 Pin 0 Data Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:1				Reserved field
DATA	0	rst_l	1'bx	RW	GPIO data. Read only when dir is set to input, read/write when dir is set to output.

1.3.3.23 GPIO Port 0 Pin 1 Data Register (0x00460008/0x0)

This is GPIO Port 0 Pin 1 Data Register. This register allows SW to read the value of pin 1 on GPIO port 0 when it is configured as an input and allows SW to write a value to pin 1 on GPIO port 0 when it is configured as an output.

Table 1-42 GPIO Port 0 Pin 1 Data Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:1				Reserved field
DATA	0	rst_l	1'bx	RW	GPIO data. Read only when dir is set to input, read/write when dir is set to output.

1.3.3.24 GPIO Port 0 Pin 2 Data Register (0x00460010/0x0)

This is GPIO Port 0 Pin 2 Data Register. This register allows SW to read the value of pin 2 on GPIO port 0 when it is configured as an input and allows SW to write a value to pin 2 on GPIO port 0 when it is configured as an output.

Table 1-43 GPIO Port 0 Pin 2 Data Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:1				Reserved field
DATA	0	rst_l	1'bx	RW	GPIO data. Read only when dir is set to input, read/write when dir is set to output.

1.3.3.25 GPIO Port 0 Pin 3 Data Register (0x00460018/0x0)

This is GPIO Port 0 Pin 3 Data Register. This register allows SW to read the value of pin 3 on GPIO port 0 when it is configured as an input and allows SW to write a value to pin 3 on GPIO port 0 when it is configured as an output.

Table 1-44 GPIO Port 0 Pin 3 Data Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:1				Reserved field
DATA	0	rst_l	1'bx	RW	GPIO data. Read only when dir is set to input, read/write when dir is set to output.

1.3.3.26 GPIO Port 0 Data Register (0x00460020/0x0)

This is GPIO Port 0 Data Register. This register allows SW to read the value of pins 0-3 on GPIO port 0 when they are configured as an input and allows SW to write a value to pins 0-3 on GPIO port 0 when they are configured as an output.

Table 1-45 GPIO Port 0 Data Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:4				Reserved field
DATA_3	3	rst_l	1'bx	RW	GPIO data 3. Read only when dir is set to input, read/write when dir is set to output.
DATA_2	2	rst_l	1'bx	RW	GPIO data 2. Read only when dir is set to input, read/write when dir is set to output.
DATA_1	1	rst_l	1'bx	RW	GPIO data 1. Read only when dir is set to input, read/write when dir is set to output.
DATA_0	0	rst_l	1'bx	RW	GPIO data 0. Read only when dir is set to input, read/write when dir is set to output.

1.3.3.27 GPIO Port 0 Control Register (0x00460028/0x0)

This is GPIO Port 0 Control Register. This register allows SW to set the direction of pins 0-3 on GPIO port 0.

Table 1-46 GPIO Port 0 Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:4				Reserved field
DIR_3	3	rst_1	0x0	RW	GPIO direction 3. 0-input, 1-output
DIR_2	2	rst_1	0x0	RW	GPIO direction 2. 0-input, 1-output
DIR_1	1	rst_1	0x0	RW	GPIO direction 1. 0-input, 1-output
DIR_0	0	rst_1	0x0	RW	GPIO direction 0. 0-input, 1-output

1.3.3.28 GPIO Port 1 Pin 0 Data Register (0x00462000/0x0)

This is GPIO Port 1 Pin 0 Data Register. This register allows SW to read the value of pin 0 on GPIO port 1 when it is configured as an input and allows SW to write a value to pin 0 on GPIO port 1 when it is configured as an output.

Table 1-47 GPIO Port 1 Pin 0 Data Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:1				Reserved field
DATA	0	rst_1	1'bx	RW	GPIO data. Read only when dir is set to input, read/write when dir is set to output.

1.3.3.29 GPIO Port 1 Pin 1 Data Register (0x00462008/0x0)

This is GPIO Port 1 Pin 1 Data Register. This register allows SW to read the value of pin 1 on GPIO port 1 when it is configured as an input and allows SW to write a value to pin 1 on GPIO port 1 when it is configured as an output.

Table 1-48 GPIO Port 1 Pin 1 Data Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:1				Reserved field
DATA	0	rst_l	1'bx	RW	GPIO data. Read only when dir is set to input, read/write when dir is set to output.

1.3.3.30 GPIO Port 1 Pin 2 Data Register (0x00462010/0x0)

This is GPIO Port 1 Pin 2 Data Register. This register allows SW to read the value of pin 2 on GPIO port 1 when it is configured as an input and allows SW to write a value to pin 2 on GPIO port 1 when it is configured as an output.

Table 1-49 GPIO Port 1 Pin 2 Data Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:1				Reserved field
DATA	0	rst_l	1'bx	RW	GPIO data. Read only when dir is set to input, read/write when dir is set to output.

1.3.3.31 GPIO Port 1 Pin 3 Data Register (0x00462018/0x0)

This is GPIO Port 1 Pin 3 Data Register. This register allows SW to read the value of pin 3 on GPIO port 1 when it is configured as an input and allows SW to write a value to pin 3 on GPIO port 1 when it is configured as an output.

Table 1-50 GPIO Port 1 Pin 3 Data Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:1				Reserved field
DATA	0	rst_l	1'bx	RW	GPIO data. Read only when dir is set to input, read/write when dir is set to output.

1.3.3.32 GPIO Port 1 Data Register (0x00462020/0x0)

This is GPIO Port 1 Data Register. This register allows SW to read the value of pins 0-3 on GPIO port 1 when they are configured as an input and allows SW to write a value to pins 0-3 on GPIO port 1 when they are configured as an output.

Table 1-51 GPIO Port 1 Data Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:4				Reserved field
DATA_3	3	rst_1	1'bx	RW	GPIO data 3. Read only when dir is set to input, read/write when dir is set to output.
DATA_2	2	rst_1	1'bx	RW	GPIO data 2. Read only when dir is set to input, read/write when dir is set to output.
DATA_1	1	rst_1	1'bx	RW	GPIO data 1. Read only when dir is set to input, read/write when dir is set to output.
DATA_0	0	rst_1	1'bx	RW	GPIO data 0. Read only when dir is set to input, read/write when dir is set to output.

1.3.3.33 GPIO Port 1 Control Register (0x00462028/0x0)

This is GPIO Port 1 Control Register. This register allows SW to set the direction of pins 0-3 on GPIO port 1.

Table 1-52 GPIO Port 1 Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:4				Reserved field
DIR_3	3	rst_1	0x0	RW	GPIO direction 3. 0-input, 1-output
DIR_2	2	rst_1	0x0	RW	GPIO direction 2. 0-input, 1-output
DIR_1	1	rst_1	0x0	RW	GPIO direction 1. 0-input, 1-output
DIR_0	0	rst_1	0x0	RW	GPIO direction 0. 0-input, 1-output

1.3.3.34 EBus EPROM Timing Control Register (0x00464000 / 0xFAD5ABF5F7)

This is the EBus EPROM Timing Control Register. This register is used to control the timing of several aspects of the EBus transaction for the EPROM chip select. This register also has enable which must be enabled before the values in this register will be used for a transaction to the EPROM. If this enable is disabled the default EPROM timing will be used.

Table 1-53 EBus EPROM Timing Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:62				Reserved field
ENABLE	61	rst_l	0x0	RW	EPROM Timing Enable, 1= Use values in this register 0 = Use the default values hard coded in the chip Use this bit override the default values and program the 6 timers with different values to make the counters faster or slower than the hard coded defaults

Table 1-53 EBus EPROM Timing Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
READY_COUNT	60:40	rst_l	0x0	RW	<p>Ready Count, The value used to program the amount of time before the EBus core will cancel a transaction to a given EBus device that is holding the ready line low to stall completion of the transaction. The counter will start at the value loaded into the read_count field and count to 0x1FFFFFF. The counter will load the value from the ready_count field when the EBus ready signal is high. When the EBus ready line is low the counter will increment. If the ready line is held low long enough for the counter to reach its maximum value of 0x1FFFFFF the EBus core will cancel the transaction.</p> <p>Example: Setting this field to 0x1FFFF8 will cause a the timer to expire after 8 clocks or 40ns.</p> <p>WARNING: DO NOT SET THIS VALUE TO CAUSE A TIMEOUT BEFORE THE STROBE COUNTER IS FINISHED.</p>

Table 1-53 EBus EPROM Timing Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
PROTOCOL_COUNT	39:32	rst_l	0xFA	RW	<p>Protocol Count, The value use to program the amount of time for 4 sections of the EBus transaction:</p> <ol style="list-style-type: none"> 1) The time EBus ale is held high 2) The time after ale is deasserted before the chip select is asserted 3) For byte stacked transactions, the minimum amount of time after chip select is deasserted before it can be asserted again if the recovery counter is programmed to a lesser value 4) For non byte-stacked back to back transactions, the minimum amount of time after chip select is de-asserted on the first transaction to the time EBus ale is asserted for the next transaction if the recovery counter is programmed to a lesser value. Note this actual amount of time between these two events can vary as the time to process the two consecutive EBus transactions can vary due to Fire's internal State. <p>The counter will start at the value loaded into the protocol_count field and count to 0xFF. The counter will load the value from the protocol_count field three separate times during the EBus transaction.</p> <p>Example: Setting this value to 0xF8 will cause the protocol time to be 8 clock cycles or 40ns.</p>

Table 1-53 EBus EPROM Timing Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
STROBE_COUNT	31:24	rst_l	0xD5	RW	<p>Strobe Count, The value used to program the amount of time that the EBus wr and EBus rd signals are asserted during an EBus transaction. The counter will start at the value loaded into the strobe_count field and count to 0xFF. The counter will load the value from the strobe_count field just before the write or read signal is asserted. When the EBus read or write line is asserted the counter will increment until it reaches its maximum value of 0xFF. When the maximum value is reached the write or read line is deasserted.</p> <p>Example: Setting this value to 0xF4 will cause the strobe time to be 12 clock cycles or 60ns.</p>

Table 1-53 EBus EPROM Timing Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RECOVERY_COUNT	23:16	rst_l	0xAB	RW	<p>Recovery Counter, The value use to program the EBus recovery time which is the minimum amount of time between two EBus transactions. Depending on the type of transaction this counter measures two possible times: 1) For byte stacked transactions, the counter is used to measure the minimum time from chip select being de-asserted on the first transaction to the time chip select is asserted on the next transaction</p> <p>2) For non byte-stacked back to back transactions, the counter is used to measure the minimum amount of time from chip select being de-asserted on the first transaction to the time EBus ale is asserted for the next transaction. Note this actual amount of time between these two events can vary as the time to process the two consecutive EBus transactions can vary due to Fire's internal State</p> <p>The counter will start at the value loaded into the recovery_count field and count to 0xFF</p> <p>Example: Setting this value to 0xF1 will cause the recovery time to be 16 clock cycles or 80ns.</p> <p>WARNING: SETTING THIS VALUE TO A VALUE WITH LESS TIME THAN PROTOCOL TIMER WILL HAVE NO EFFECT ON THE EBus TRANSACTION.</p>

Table 1-53 EBus EPROM Timing Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
HOLD_COUNT	15:8	rst_l	0xF5	RW	<p>Hold Counter, The value used to program the amount of time between the deassertion of the EBus wr or EBus rd signals and the deassertion of chip select. The counter will start at the value loaded into the hold_count field and count to 0xFF. The counter will load the value from the hold_count field just before the write or read signal is deasserted. The counter will then increment until it reaches it maximum value of 0xFF. When the maximum value is reached the chip select is deasserted.</p> <p>Example: Setting this value to 0xFA will cause the strobe time to be 6 clock cycles or 30ns.</p>
SETUP_COUNT	7:0	rst_l	0xF7	RW	<p>Setup Counter, The value used to program the amount of time plus 2 cycles between the assertion of chip select and the assertion of the EBus wr or EBus rd signals. The counter will start at the value loaded into the setup_count field and count to 0xFF. The counter will load the value from the setup_count field just before the chip select is deasserted. The counter will then increment until it reaches it maximum value of 0xFF. When the maximum value is reached the EBus rd or EBus write signal is asserted.</p> <p>Example: Setting this value to 0xFA will cause the strobe time to be 8 clock cycles or 40ns.</p>

1.3.3.35 EBus Chip Select 1 Timing Control Register (0x00464008 / 0xFAD5ABF5F7)

This is the EBus Chip Select 1 Timing Control Register. This register is used to control the timing of several aspects of the EBus transaction for chip select 1. Unlike the EPROM chip select the values in this register will always be used for an EBus transaction to this chip select.

Table 1-54 EBus Chip Select 1 Timing Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:61				Reserved field
READY_COUNT	60:40	rst_1	0x0	RW	<p>Ready Count, The value used to program the amount of time before the EBus core will cancel a transaction to a given EBus device that is holding the ready line low to stall completion of the transaction. The counter will start at the value loaded into the read_count field and count to 0x1FFFFFF. The counter will load the value from the ready_count field when the EBus ready signal is high. When the EBus ready line is low the counter will increment. If the ready line is held low long enough for the counter to reach its maximum value of 0x1FFFFFF the EBus core will cancel the transaction.</p> <p>Example: Setting this field to 0x1FFFF8 will cause the timer to expire after 8 clocks or 40ns.</p> <p>WARNING: DO NOT SET THIS VALUE TO CAUSE A TIMEOUT BEFORE THE STROBE COUNTER IS FINISHED.</p>

Table 1-54 EBus Chip Select 1 Timing Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
PROTOCOL_COUNT	39:32	rst_l	0xFA	RW	<p>Protocol Count, The value use to program the amount of time for 4 sections of the EBus transaction:</p> <ol style="list-style-type: none"> 1) The time EBus ale is held high 2) The time after ale is deasserted before the chip select is asserted 3) For byte stacked transactions, the minimum amount of time after chip select is deasserted before it can be asserted again if the recovery counter is programmed to a lesser value 4) For non byte-stacked back to back transactions, the minimum amount of time after chip select is de-asserted on the first transaction to the time EBus ale is asserted for the next transaction if the recovery counter is programmed to a lesser value. Note this actual amount of time between these two events can vary as the time to process the two consecutive EBus transactions can vary due to Fire's internal State. <p>The counter will start at the value loaded into the protocol_count field and count to 0xFF. The counter will load the value from the protocol_count field three separate times during the EBus transaction.</p> <p>Example: Setting this value to 0xF8 will cause the protocol time to be 8 clock cycles or 40ns.</p>

Table 1-54 EBus Chip Select 1 Timing Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
STROBE_COUNT	31:24	rst_l	0xD5	RW	<p>Strobe Count, The value used to program the amount of time that the EBus wr and EBus rd signals are asserted during an EBus transaction. The counter will start at the value loaded into the strobe_count field and count to 0xFF. The counter will load the value from the strobe_count field just before the write or read signal is asserted. When the EBus read or write line is asserted the counter will increment until it reaches its maximum value of 0xFF. When the maximum value is reached the write or read line is deasserted.</p> <p>Example: Setting this value to 0xF4 will cause the strobe time to be 12 clock cycles or 60ns.</p>

Table 1-54 EBus Chip Select 1 Timing Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RECOVERY_COUNT	23:16	rst_l	0xAB	RW	<p>Recovery Counter, The value use to program the EBus recovery time which is the minimum amount of time between two EBus transactions. Depending on the type of transaction this counter measures two possible times: 1) For byte stacked transactions, the counter is used to measure the minimum time from chip select being de-asserted on the first transaction to the time chip select is asserted on the next transaction</p> <p>2) For non byte-stacked back to back transactions, the counter is used to measure the minimum amount of time from chip select being de-asserted on the first transaction to the time EBus ale is asserted for the next transaction. Note this actual amount of time between these two events can vary as the time to process the two consecutive EBus transactions can vary due to Fire's internal State</p> <p>The counter will start at the value loaded into the recovery_count field and count to 0xFF</p> <p>Example: Setting this value to 0xF1 will cause the recovery time to be 16 clock cycles or 80ns.</p> <p>WARNING: SETTING THIS VALUE TO A VALUE WITH LESS TIME THAN PROTOCOL TIMER WILL HAVE NO EFFECT ON THE EBus TRANSACTION.</p>

Table 1-54 EBus Chip Select 1 Timing Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
HOLD_COUNT	15:8	rst_l	0xF5	RW	<p>Hold Counter, The value used to program the amount of time between the deassertion of the EBus wr or EBus rd signals and the deassertion of chip select. The counter will start at the value loaded into the hold_count field and count to 0xFF. The counter will load the value from the hold_count field just before the write or read signal is deasserted. The counter will then increment until it reaches it maximum value of 0xFF. When the maximum value is reached the chip select is deasserted.</p> <p>Example: Setting this value to 0xFA will cause the strobe time to be 6 clock cycles or 30ns.</p>
SETUP_COUNT	7:0	rst_l	0xF7	RW	<p>Setup Counter, The value used to program the amount of time plus 2 cycles between the assertion of chip select and the assertion of the EBus wr or EBus rd signals. The counter will start at the value loaded into the setup_count field and count to 0xFF. The counter will load the value from the setup_count field just before the chip select is deasserted. The counter will then increment until it reaches it maximum value of 0xFF. When the maximum value is reached the EBus rd or EBus write signal is asserted.</p> <p>Example: Setting this value to 0xFA will cause the strobe time to be 8 clock cycles or 40ns.</p>

1.3.3.36 EBus Chip Select 2 Timing Control Register (0x00464010 / 0xFAD5ABF5F7)

This is the EBus Chip Select 2 Timing Control Register. This register is used to control the timing of several aspects of the EBus transaction for chip select 2. Unlike the EPROM chip select the values in this register will always be used for an EBus transaction to this chip select.

Table 1-55 EBus Chip Select 2 Timing Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:61				Reserved field
READY_COUNT	60:40	rst_1	0x0	RW	<p>Ready Count, The value used to program the amount of time before the EBus core will cancel a transaction to a given EBus device that is holding the ready line low to stall completion of the transaction. The counter will start at the value loaded into the read_count field and count to 0x1FFFFFF. The counter will load the value from the ready_count field when the EBus ready signal is high. When the EBus ready line is low the counter will increment. If the ready line is held low long enough for the counter to reach its maximum value of 0x1FFFFFF the EBus core will cancel the transaction.</p> <p>Example: Setting this field to 0x1FFFF8 will cause the timer to expire after 8 clocks or 40ns.</p> <p>WARNING: DO NOT SET THIS VALUE TO CAUSE A TIMEOUT BEFORE THE STROBE COUNTER IS FINISHED.</p>

Table 1-55 EBus Chip Select 2 Timing Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
PROTOCOL_COUNT	39:32	rst_l	0xFA	RW	<p>Protocol Count, The value use to program the amount of time for 4 sections of the EBus transaction:</p> <ol style="list-style-type: none"> 1) The time EBus ale is held high 2) The time after ale is deasserted before the chip select is asserted 3) For byte stacked transactions, the minimum amount of time after chip select is deasserted before it can be asserted again if the recovery counter is programmed to a lesser value 4) For non byte-stacked back to back transactions, the minimum amount of time after chip select is de-asserted on the first transaction to the time EBus ale is asserted for the next transaction if the recovery counter is programmed to a lesser value. Note this actual amount of time between these two events can vary as the time to process the two consecutive EBus transactions can vary due to Fire's internal State. <p>The counter will start at the value loaded into the protocol_count field and count to 0xFF. The counter will load the value from the protocol_count field three separate times during the EBus transaction.</p> <p>Example: Setting this value to 0xF8 will cause the protocol time to be 8 clock cycles or 40ns.</p>

Table 1-55 EBus Chip Select 2 Timing Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
STROBE_COUNT	31:24	rst_l	0xD5	RW	<p>Strobe Count, The value used to program the amount of time that the EBus wr and EBus rd signals are asserted during an EBus transaction. The counter will start at the value loaded into the strobe_count field and count to 0xFF. The counter will load the value from the strobe_count field just before the write or read signal is asserted. When the EBus read or write line is asserted the counter will increment until it reaches its maximum value of 0xFF. When the maximum value is reached the write or read line is deasserted.</p> <p>Example: Setting this value to 0xF4 will cause the strobe time to be 12 clock cycles or 60ns.</p>

Table 1-55 EBus Chip Select 2 Timing Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RECOVERY_COUNT	23:16	rst_l	0xAB	RW	<p>Recovery Counter, The value use to program the EBus recovery time which is the minimum amount of time between two EBus transactions. Depending on the type of transaction this counter measures two possible times: 1) For byte stacked transactions, the counter is used to measure the minimum time from chip select being de-asserted on the first transaction to the time chip select is asserted on the next transaction</p> <p>2) For non byte-stacked back to back transactions, the counter is used to measure the minimum amount of time from chip select being de-asserted on the first transaction to the time EBus ale is asserted for the next transaction. Note this actual amount of time between these two events can vary as the time to process the two consecutive EBus transactions can vary due to Fire's internal State</p> <p>The counter will start at the value loaded into the recovery_count field and count to 0xFF</p> <p>Example: Setting this value to 0xF1 will cause the recovery time to be 16 clock cycles or 80ns.</p> <p>WARNING: SETTING THIS VALUE TO A VALUE WITH LESS TIME THAN PROTOCOL TIMER WILL HAVE NO EFFECT ON THE EBus TRANSACTION.</p>

Table 1-55 EBus Chip Select 2 Timing Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
HOLD_COUNT	15:8	rst_l	0xF5	RW	<p>Hold Counter, The value used to program the amount of time between the deassertion of the EBus wr or EBus rd signals and the deassertion of chip select. The counter will start at the value loaded into the hold_count field and count to 0xFF. The counter will load the value from the hold_count field just before the write or read signal is deasserted. The counter will then increment until it reaches it maximum value of 0xFF. When the maximum value is reached the chip select is deasserted.</p> <p>Example: Setting this value to 0xFA will cause the strobe time to be 6 clock cycles or 30ns.</p>
SETUP_COUNT	7:0	rst_l	0xF7	RW	<p>Setup Counter, The value used to program the amount of time plus 2 cycles between the assertion of chip select and the assertion of the EBus wr or EBus rd signals. The counter will start at the value loaded into the setup_count field and count to 0xFF. The counter will load the value from the setup_count field just before the chip select is deasserted. The counter will then increment until it reaches it maximum value of 0xFF. When the maximum value is reached the EBus rd or EBus write signal is asserted.</p> <p>Example: Setting this value to 0xFA will cause the strobe time to be 8 clock cycles or 40ns.</p>

1.3.3.37 EBus Chip Select 3 Timing Control Register (0x00464018 / 0xFAD5ABF5F7)

This is the EBus Chip Select 3 Timing Control Register. This register is used to control the timing of several aspects of the EBus transaction for chip select 3. Unlike the EPROM chip select the values in this register will always be used for an EBus transaction to this chip select.

Table 1-56 EBus Chip Select 3 Timing Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:61				Reserved field
READY_COUNT	60:40	rst_1	0x0	RW	<p>Ready Count, The value used to program the amount of time before the EBus core will cancel a transaction to a given EBus device that is holding the ready line low to stall completion of the transaction. The counter will start at the value loaded into the read_count field and count to 0x1FFFFFF. The counter will load the value from the ready_count field when the EBus ready signal is high. When the EBus ready line is low the counter will increment. If the ready line is held low long enough for the counter to reach its maximum value of 0x1FFFFFF the EBus core will cancel the transaction.</p> <p>Example: Setting this field to 0x1FFFF8 will cause the timer to expire after 8 clocks or 40ns.</p> <p>WARNING: DO NOT SET THIS VALUE TO CAUSE A TIMEOUT BEFORE THE STROBE COUNTER IS FINISHED.</p>

Table 1-56 EBus Chip Select 3 Timing Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
PROTOCOL_COUNT	39:32	rst_l	0xFA	RW	<p>Protocol Count, The value use to program the amount of time for 4 sections of the EBus transaction:</p> <ol style="list-style-type: none"> 1) The time EBus ale is held high 2) The time after ale is deasserted before the chip select is asserted 3) For byte stacked transactions, the minimum amount of time after chip select is deasserted before it can be asserted again if the recovery counter is programmed to a lesser value 4) For non byte-stacked back to back transactions, the minimum amount of time after chip select is de-asserted on the first transaction to the time EBus ale is asserted for the next transaction if the recovery counter is programmed to a lesser value. Note this actual amount of time between these two events can vary as the time to process the two consecutive EBus transactions can vary due to Fire's internal State. <p>The counter will start at the value loaded into the protocol_count field and count to 0xFF. The counter will load the value from the protocol_count field three separate times during the EBus transaction.</p> <p>Example: Setting this value to 0xF8 will cause the protocol time to be 8 clock cycles or 40ns.</p>

Table 1-56 EBus Chip Select 3 Timing Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
STROBE_COUNT	31:24	rst_l	0xD5	RW	<p>Strobe Count, The value used to program the amount of time that the EBus wr and EBus rd signals are asserted during an EBus transaction. The counter will start at the value loaded into the strobe_count field and count to 0xFF. The counter will load the value from the strobe_count field just before the write or read signal is asserted. When the EBus read or write line is asserted the counter will increment until it reaches its maximum value of 0xFF. When the maximum value is reached the write or read line is deasserted.</p> <p>Example: Setting this value to 0xF4 will cause the strobe time to be 12 clock cycles or 60ns.</p>

Table 1-56 EBus Chip Select 3 Timing Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RECOVERY_COUNT	23:16	rst_l	0xAB	RW	<p>Recovery Counter, The value use to program the EBus recovery time which is the minimum amount of time between two EBus transactions. Depending on the type of transaction this counter measures two possible times: 1) For byte stacked transactions, the counter is used to measure the minimum time from chip select being de-asserted on the first transaction to the time chip select is asserted on the next transaction</p> <p>2) For non byte-stacked back to back transactions, the counter is used to measure the minimum amount of time from chip select being de-asserted on the first transaction to the time EBus ale is asserted for the next transaction. Note this actual amount of time between these two events can vary as the time to process the two consecutive EBus transactions can vary due to Fire's internal State</p> <p>The counter will start at the value loaded into the recovery_count field and count to 0xFF</p> <p>Example: Setting this value to 0xF1 will cause the recovery time to be 16 clock cycles or 80ns.</p> <p>WARNING: SETTING THIS VALUE TO A VALUE WITH LESS TIME THAN PROTOCOL TIMER WILL HAVE NO EFFECT ON THE EBus TRANSACTION.</p>

Table 1-56 EBus Chip Select 3 Timing Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
HOLD_COUNT	15:8	rst_l	0xF5	RW	<p>Hold Counter, The value used to program the amount of time between the deassertion of the EBus wr or EBus rd signals and the deassertion of chip select. The counter will start at the value loaded into the hold_count field and count to 0xFF. The counter will load the value from the hold_count field just before the write or read signal is deasserted. The counter will then increment until it reaches it maximum value of 0xFF. When the maximum value is reached the chip select is deasserted.</p> <p>Example: Setting this value to 0xFA will cause the strobe time to be 6 clock cycles or 30ns.</p>
SETUP_COUNT	7:0	rst_l	0xF7	RW	<p>Setup Counter, The value used to program the amount of time plus 2 cycles between the assertion of chip select and the assertion of the EBus wr or EBus rd signals. The counter will start at the value loaded into the setup_count field and count to 0xFF. The counter will load the value from the setup_count field just before the chip select is deasserted. The counter will then increment until it reaches it maximum value of 0xFF. When the maximum value is reached the EBus rd or EBus write signal is asserted.</p> <p>Example: Setting this value to 0xFA will cause the strobe time to be 8 clock cycles or 40ns.</p>

1.3.3.38 I2C 0 Input Monitor Register (0x00466000/0x0)

This is the I2C 0 Input Monitor Register. This register allows SW to be able to read the value on the clock and data line for I2C core 0. These values will be the values from the actual pins of the chip and not the values of the corresponding data and clock drive registers

Table 1-57 I2C 0 Input Monitor Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:2				Reserved field
SDC	1	rst_l	1'bx	R	Used store the incoming value of the Clock line for the I2C core 0. This register constantly is updated with the value being seen on the bus.
SDA	0	rst_l	1'bx	R	Used store the incoming value of the Data line for the I2C core 0. This register constantly is updated with the value being seen on the bus.

1.3.3.39 I2C 0 Data Drive Register (0x00466008/0x1)

This is the I2C 0 Data Drive Register. This register allows SW to be able to override the data out pin for I2C core 0. This register's value when set to 0 will be used and passed to the I2C data pin of the chip ignoring the actual output value from the I2C core. When reading this register you will be reading the value programming into the flop and not the actual value of the data pin on the chip.

Table 1-58 I2C 0 Data Drive Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:1				Reserved field
SDA	0	rst_l	0x1	RW	Used to drive the output value of the data line for the I2C core 0. If it is set to a value of 1, this will have no effect on the Data line output. If this set to a value of 0, a value of zero will be driven on the bus.

1.3.3.40 I2C 0 Clock Drive Register (0x00466010 / 0x1)

This is the I2C 0 Clock Drive Register. This register allows SW to be able to override the clock out pin for I2C core 0. This register's value when set to 0 will be used and passed to the I2C clock pin of the chip ignoring the actual output value from the I2C core. When reading this register you will be reading the value programming into the flop and not the actual value of the clk pin on the chip.

Table 1-59 I2C 0 Clock Drive Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:1				Reserved field
SCL	0	rst_1	0x1	RW	Used to drive the output value of the clock line for the I2C core 0. If it is set to a value of 1, this will have no effect on the clock line output. If this set to a value of 0, a value of zero will be driven on the bus.

1.3.3.41 I2C 1 Input Monitor Register (0x00468000 / 0x0)

This is the I2C 1 Input Monitor Register. This register allows SW to be able to read the value on the clock and data line for I2C core 1. These values will be the values from the actual pins of the chip and not the values of the corresponding data and clock drive registers

Table 1-60 I2C 1 Input Monitor Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:2				Reserved field
SDC	1	rst_1	1'bx	R	Used store the incoming value of the Clock line for the I2C core 1. This register constantly is updated with the value being seen on the bus.
SDA	0	rst_1	1'bx	R	Used store the incoming value of the Data line for the I2C core 1. This register constantly is updated with the value being seen on the bus.

1.3.3.42 I2C 1 Data Drive Register (0x00468008/0x1)

This is the I2C 1 Data Drive Register. This register allows SW to be able to override the data out pin for I2C core 1. This register's value when set to 0 will be used and passed to the I2C data pin of the chip ignoring the actual output value from the I2C core. When reading this register you will be reading the value programming into the flop and not the actual value of the data pin on the chip.

Table 1-61 I2C 1 Data Drive Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:1				Reserved field
SDA	0	rst_1	0x1	RW	Used to drive the output value of the data line for the I2C core 1. If it is set to a value of 1, this will have no effect on the Data line output. If this set to a value of 0, a value of zero will be driven on the bus.

1.3.3.43 I2C 1 Clock Drive Register (0x00468010/0x1)

This is the I2C 1 Clock Drive Register. This register allows SW to be able to override the clock out pin for I2C core 1. This register's value when set to 0 will be used and passed to the I2C clock pin of the chip ignoring the actual output value from the I2C core. When reading this register you will be reading the value programming into the flop and not the actual value of the clk pin on the chip.

Table 1-62 I2C 1 Clock Drive Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:1				Reserved field
SCL	0	rst_1	0x1	RW	Used to drive the output value of the clock line for the I2C core 1. If it is set to a value of 1, this will have no effect on the clock line output. If this set to a value of 0, a value of zero will be driven on the bus.

1.3.3.44 PCIE-A Leaf CSR Ring Slow Only Access (0x00470000 / 0x0)

This is the PCIE-A Leaf CSR Ring Slow Only Access Register. This register gives SW the ability to override the Fast and Medium CSR rings in the PCIE-A leaf and force every internal CSR to the 8 MB region to use the slow path. This should only be used in the lab or if there is a bug in the design which prevents those rings from functioning properly. This will cause a drastic performance loss if this register is used.

Table 1-63 PCIE-A Leaf CSR Ring Slow Only Access

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:1				Reserved field
SLOW_ONLY	0	rst_1	0x0	RW	Used to default all CSR Ring access to slow only mode instead of fast and medium.

1.3.3.45 PCIE-B Leaf CSR Ring Slow Only Access (0x00470008 / 0x0)

This is the PCIE-B Leaf CSR Ring Slow Only Access Register. This register gives SW the ability to override the Fast and Medium CSR rings in the PCIE-A leaf and force every internal CSR to the 8 MB region to use the slow path. This should only be used in the lab or if there is a bug in the design which prevents those rings from functioning properly. This will cause a drastic performance loss if this register is used.

Table 1-64 PCIE-B Leaf CSR Ring Slow Only Access

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:1				Reserved field
SLOW_ONLY	0	rst_1	0x0	RW	Used to default all CSR Ring access to slow only mode instead of fast and medium.

1.3.3.46 JBus Parity Control Register (0x00470010 / 0x0)

This is the JBus Parity Control Register. This register controls Fire's detection of JBus Control and Data parity, and assertion of associated JBUS parity error interrupts. In order for Fire to perform parity checking the enable bit in this register must be enabled. This register is also used to inject parity errors on to the JBus on the next address or data cycle.

Table 1-65 JBus Parity Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
P_EN	63	rst_1	0x0	RW	Parity Check Enable. Parity is checked on incoming address/data when set to one. Parity is always generated on outgoing address/data.
RESERVED	62:6				Reserved field
INVERT_PAR	5:2	rst_1	0x0	RW	A one in bit position X will invert J_ADP[x], on the next cycle as determined by Next_addr or Next_data. Note that J_ADP[3] also covers the AD_TYPE lines, and could be interpreted as a control error as well as a data error.
NEXT_DATA	1	rst_1	0x0	RW1S	When set by Software, hardware will invert J_ADP bits (as selected by Invert_Par) on the next Data cycle that Fire sources (Read Return data, a write data cycle, Interrupt vector). After the parity is inverted, the Next_data bit will be cleared. If both Next_data and Next_Addr are set, the next cycle Fire sources will have the J_ADP bit inverted.

Table 1-65 JBus Parity Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
NEXT_ADDR	0	rst_l	0x0	RW1S	When set by Software, hardware will invert J_ADP bits (as selected by Invert_Par) on the next Address cycle that Fire sources (RDO, RDS, RDD, INT, WRB, WRBC). After the parity is inverted, the Next_data bit will be cleared.If both Next_data and Next_Addr are set, the next cycle Fire sources will have the J_ADP bit inverted.

1.3.3.47 JBus Scratch Register 1 (0x00470018/0x0)

This is the JBus Scratch Register 1. This register is used by SW as a scratch pad to store data values. This register serves no hardware function.

Table 1-66 JBus Scratch Register 1

Field	Bits	Reset Name	Reset Value	Type	Description
DATA	63:0	rst_l	0x0	RW	This is a 64bit scratch register ONLY used by software. It has no effect on ANY hardware function. Software may read and write this as they see fit. The values will NOT persist across soft reset

1.3.3.48 JBus Scratch Register 2 (0x00470020/0x0)

This is the JBus Scratch Register 2. This register is used by SW as a scratch pad to store data values. This register serves no hardware function.

Table 1-67 JBus Scratch Register 2

Field	Bits	Reset Name	Reset Value	Type	Description
DATA	63:0	rst_l	0x0	RW	This is a 64bit scratch register ONLY used by software. It has no effect on ANY hardware function. Software may read and write this as they see fit. The values will NOT persist across soft reset

1.3.3.49 JBC Error Logic Analyzer Trigger Enable for J_ERR (0x00470028 / 0x3FFFFFFF1FFFFFFF)

This is the JBus J_ERR Logic Analyzer Trigger. This register is used to enable JBC errors to cause the J_ERR pin to transition from low to high on the detection of an error. A value of 1 written to any bit in the register will cause the transition in J_ERR if that type of error is detected by the JBC core. A value of 0 will prevent J_ERR from transition. By default all of the JBC errors are enabled to cause the trigger of J_ERR.

Table 1-68 JBC Error Logic Analyzer Trigger Enable for J_ERR

Field	Bits	Reset Name	Reset Value	Type	Description
SPARE_S_INT_EN	63:61	por_1	0x1	RW	Spare Errors, Primary J_ERR Logic Analyzer Trigger EnableBit
PIO_UNMAP_RD_S_INT_EN	60	por_1	0x1	RW	NCWR or NCBWR Unmapped, Secondary J_ERR Logic Analyzer Trigger EnableBit
ILL_ACC_RD_S_INT_EN	59	por_1	0x1	RW	Illegal NCRD or NCBRD access, Secondary J_ERR Logic Analyzer Trigger EnableBit
EBUS_TO_S_LOG_EN	58	por_1	0x1	RW	EBus Ready Timeout Detected, Secondary J_ERR Logic Analyzer Trigger EnableBit
MB_PEA_S_INT_EN	57	por_1	0x1	RW	Merge Buffer Address Parity Error Detected, Secondary J_ERR Logic Analyzer Trigger EnableBit
MB_PER_S_INT_EN	56	por_1	0x1	RW	Merge Buffer Parity Error Detected on read, Secondary J_ERR Logic Analyzer Trigger EnableBit
MB_PEW_S_INT_EN	55	por_1	0x1	RW	Merge Buffer Parity Error Detected on write, Secondary J_ERR Logic Analyzer Trigger EnableBit
UE_ASYNC_S_INT_EN	54	por_1	0x1	RW	UE Asynchronous Fault, Secondary J_ERR Logic Analyzer Trigger EnableBit
CE_ASYNC_S_INT_EN	53	por_1	0x1	RW	CE Asynchronous Fault, Secondary J_ERR Logic Analyzer Trigger EnableBit

Table 1-68 JBC Error Logic Analyzer Trigger Enable for J_ERR

Field	Bits	Reset Name	Reset Value	Type	Description
JTE_S_INT_EN	52	por_1	0x1	RW	JBus Time-out Err, Secondary J_ERR Logic Analyzer Trigger EnableBit
JBE_S_INT_EN	51	por_1	0x1	RW	JBus Bus Error, Secondary J_ERR Logic Analyzer Trigger EnableBit
JUE_S_INT_EN	50	por_1	0x1	RW	JBus Unmapped Error, Secondary J_ERR Logic Analyzer Trigger EnableBit
IJP_S_INT_EN	49	por_1	0x1	RW	Invalid JBUS Port Error, Secondary J_ERR Logic Analyzer Trigger EnableBit
ICISE_S_INT_EN	48	por_1	0x1	RW	JBus Illegal Coherency Install State, Secondary J_ERR Logic Analyzer Trigger EnableBit
CPE_S_INT_EN	47	por_1	0x1	RW	Control Parity Error, Secondary J_ERR Logic Analyzer Trigger EnableBit
APE_S_INT_EN	46	por_1	0x1	RW	Address Parity Error, Secondary J_ERR Logic Analyzer Trigger EnableBit
WR_DPE_S_INT_EN	45	por_1	0x1	RW	Write Data Parity, Secondary J_ERR Logic Analyzer Trigger EnableBit
RD_DPE_S_INT_EN	44	por_1	0x1	RW	Read Data Parity Error, Secondary J_ERR Logic Analyzer Trigger EnableBit.
ILL_BMW_S_INT_EN	43	por_1	0x1	RW	Illegal byte mask for NCWR, Secondary J_ERR Logic Analyzer Trigger EnableBit
ILL_BMR_S_INT_EN	42	por_1	0x1	RW	Illegal byte mask for NCRD, Secondary J_ERR Logic Analyzer Trigger EnableBit
BJC_S_INT_EN	41	por_1	0x1	RW	Bad JBus cmd, Secondary J_ERR Logic Analyzer Trigger EnableBit

Table 1-68 JBC Error Logic Analyzer Trigger Enable for J_ERR

Field	Bits	Reset Name	Reset Value	Type	Description
PIO_UNMAP_S_INT_EN	40	por_1	0x1	RW	NCWR or NCBWR Unmapped, Secondary J_ERR Logic Analyzer Trigger EnableBit
PIO_DPE_S_INT_EN	39	por_1	0x1	RW	PIO Data Parity Error, Secondary J_ERR Logic Analyzer Trigger EnableBit
PIO_CPE_S_INT_EN	38	por_1	0x1	RW	PIO Command Parity Error, Secondary J_ERR Logic Analyzer Trigger EnableBit
ILL_ACC_S_INT_EN	37	por_1	0x1	RW	Illegal NCWR or NCBWR access, Secondary J_ERR Logic Analyzer Trigger EnableBit
UNSOL_RD_S_INT_EN	36	por_1	0x1	RW	Unsolicited read response, Secondary J_ERR Logic Analyzer Trigger EnableBit
UNSOL_INTR_S_INT_EN	35	por_1	0x1	RW	Unsolicited interrupt ACK/NAK, Secondary J_ERR Logic Analyzer Trigger EnableBit
JTCEEW_S_INT_EN	34	por_1	0x1	RW	JBus Time-out on write error, Secondary J_ERR Logic Analyzer Trigger EnableBit
JTCEEI_S_INT_EN	33	por_1	0x1	RW	JBus Time-out on interrupt error, Secondary J_ERR Logic Analyzer Trigger EnableBit
JTCEER_S_INT_EN	32	por_1	0x1	RW	JBus Time-out on read error, Secondary J_ERR Logic Analyzer Trigger EnableBit
SPARE_P_INT_EN	31:29	por_1	0x0	RW	Spare Errors, Primary J_ERR Logic Analyzer Trigger EnableBit
PIO_UNMAP_RD_P_INT_EN	28	por_1	0x1	RW	NCWR or NCBWR Unmapped, Primary J_ERR Logic Analyzer Trigger EnableBit

Table 1-68 JBC Error Logic Analyzer Trigger Enable for J_ERR

Field	Bits	Reset Name	Reset Value	Type	Description
ILL_ACC_RD_P_INT_EN	27	por_1	0x1	RW	Illegal NCRD or NCBRD access, Primary J_ERR Logic Analyzer Trigger EnableBit
EBUS_TO_P_LOG_EN	26	por_1	0x1	RW	EBus Ready Timeout Detected, Primary J_ERR Logic Analyzer Trigger EnableBit
MB_PEA_P_INT_EN	25	por_1	0x1	RW	Merge Buffer Address Parity Error Detected, Primary J_ERR Logic Analyzer Trigger EnableBit
MB_PER_P_INT_EN	24	por_1	0x1	RW	Merge Buffer Parity Error Detected on read, Primary J_ERR Logic Analyzer Trigger EnableBit
MB_PEW_P_INT_EN	23	por_1	0x1	RW	Merge Buffer Parity Error Detected on write, Primary J_ERR Logic Analyzer Trigger EnableBit
UE_ASYNC_P_INT_EN	22	por_1	0x1	RW	UE Asynchronous Fault, Primary J_ERR Logic Analyzer Trigger EnableBit
CE_ASYNC_P_INT_EN	21	por_1	0x1	RW	CE Asynchronous Fault, Primary J_ERR Logic Analyzer Trigger EnableBit
JTE_P_INT_EN	20	por_1	0x1	RW	JBus Time-out Err, Primary J_ERR Logic Analyzer Trigger EnableBit
JBE_P_INT_EN	19	por_1	0x1	RW	JBus Bus Error, Primary J_ERR Logic Analyzer Trigger EnableBit
JUE_P_INT_EN	18	por_1	0x1	RW	JBus Unmapped Error, Primary J_ERR Logic Analyzer Trigger EnableBit
IJP_P_INT_EN	17	por_1	0x1	RW	Invalid JBUS Port Error, Primary J_ERR Logic Analyzer Trigger EnableBit

Table 1-68 JBC Error Logic Analyzer Trigger Enable for J_ERR

Field	Bits	Reset Name	Reset Value	Type	Description
ICISE_P_INT_EN	16	por_1	0x1	RW	JBus Illegal Coherency Install State, Primary J_ERR Logic Analyzer Trigger EnableBit
CPE_P_INT_EN	15	por_1	0x1	RW	Control Parity Error, Primary J_ERR Logic Analyzer Trigger EnableBit
APE_P_INT_EN	14	por_1	0x1	RW	Address Parity Error, Primary J_ERR Logic Analyzer Trigger EnableBit
WR_DPE_P_INT_EN	13	por_1	0x1	RW	Write Data Parity, Primary J_ERR Logic Analyzer Trigger EnableBit
RD_DPE_P_INT_EN	12	por_1	0x1	RW	Read Data Parity Error, Primary J_ERR Logic Analyzer Trigger EnableBit.
ILL_BMW_P_INT_EN	11	por_1	0x1	RW	Illegal byte mask for NCWR, Primary J_ERR Logic Analyzer Trigger EnableBit
ILL_BMR_P_INT_EN	10	por_1	0x1	RW	Illegal byte mask for NCRD, Primary J_ERR Logic Analyzer Trigger EnableBit
BJC_P_INT_EN	9	por_1	0x1	RW	Bad JBus cmd, Primary J_ERR Logic Analyzer Trigger EnableBit
PIO_UNMAP_P_INT_EN	8	por_1	0x1	RW	NCWR or NCBWR Unmapped, Primary J_ERR Logic Analyzer Trigger EnableBit
PIO_DPE_P_INT_EN	7	por_1	0x1	RW	PIO Data Parity Error, Primary J_ERR Logic Analyzer Trigger EnableBit
PIO_CPE_P_INT_EN	6	por_1	0x1	RW	PIO Command Parity Error, Primary J_ERR Logic Analyzer Trigger EnableBit
ILL_ACC_P_INT_EN	5	por_1	0x1	RW	Illegal NCWR or NCBWR access, Primary J_ERR Logic Analyzer Trigger EnableBit

Table 1-68 JBC Error Logic Analyzer Trigger Enable for J_ERR

Field	Bits	Reset Name	Reset Value	Type	Description
UNSOL_RD_P_INT_EN	4	por_1	0x1	RW	Unsolicited read response, Primary J_ERR Logic Analyzer Trigger EnableBit
UNSOL_INTR_P_INT_EN	3	por_1	0x1	RW	Unsolicited interrupt ACK/NAK, Primary J_ERR Logic Analyzer Trigger EnableBit
JTCEEW_P_INT_EN	2	por_1	0x1	RW	JBus Time-out on write error, Primary J_ERR Logic Analyzer Trigger EnableBit
JTCEEI_P_INT_EN	1	por_1	0x1	RW	JBus Time-out on interrupt error, Primary J_ERR Logic Analyzer Trigger EnableBit
JTCEER_P_INT_EN	0	por_1	0x1	RW	JBus Time-out on read error, Primary J_ERR Logic Analyzer Trigger EnableBit

1.3.3.50 JBus Scratch Persistent Register (0x00470030/0x0)

This is the JBus Scratch Persistent Register. This register is used by SW as a scratch pad to store data values across soft reset. This register serves no hardware function.

Table 1-69 JBus Scratch Persistent Register

Field	Bits	Reset Name	Reset Value	Type	Description
DATA	63:0	por_1	0x0	RW	This is a 64bit scratch register ONLY used by software. It has no effect on ANY hardware function. Software may read and write this as they see fit. The values WILL persist across soft reset

1.3.3.51 JBC Error Log Enable Register (0x00471000/0x1FFFFFFF)

This is the JBC Error Log Enable Register. It is used to enable the logging for all of the possible JBC errors detected by Fire. By default all of the errors are enabled to be logged.

This and the following JBC error registers have chosen to use the Error Grouping Feature. This allows more than 1 error to be grouped together to the same error logging register.

The JBC has the following Error Groups: A Fatal Error Group, a DMCINT ODCD Group, DMCINT IDC Group, JBUSINT In Group, JBUSINT Out Group, Merge Group and the CSR Group.

The Errors associated with each group are as follows:

Fatal Error Group: mb_pea, cpe, ape, jtceei, jtceer, jtceew, pio_cpe, spare bit 0 and 1.

DMCINT ODCD Group: pio_unmap, pio_unmap_rd, pio_dpe, ill_acc, ill_acc_rd, spare bit 2.

DMCINT IDC Group: unsol_rd, unsol_intr.

JBUSINT In Group: ue_asyn, ce_asyn, jte, jbe, jue, icise, wr_dpe, rd_dpe, ill_bmw, ill_bmr, bjc.

JBUSINT Out Group: ijp.

Merge Group: mb_per, mb_pew.

CSR Group: ebus_to.

Note: All parity errors detected by the JBC will be logged on a per JBus cycle basis, not a per transaction basis. Thus, if a single transaction contains multiple parity errors, they will each be processed individually.

Note: For all error groups with the exception of the Fatal Error Group, if multiple primary errors are detected simultaneously the associated data stored for those errors corresponds to all primary errors logged. For the Fatal error group, there is a priority order for the errors. If multiple primary fatal errors are detected simultaneously the data associated with the highest priority error will be stored. The priority order for the fatal errors is as follows: mb_pea, cpe, ape, jtceer, jtceei, and pio_cpe. All other fatal errors don't store error information.

Table 1-70 JBC Error Log Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
SPARE_LOG_EN	31:29	por_1	0x0	RW	Spare Error, Error Log Enable Bit
PIO_UNMAP_RD_LOG_EN	28	por_1	0x1	RW	NCRD or NCBRD Unmapped, Error Log Enable Bit
ILL_ACC_RD_LOG_EN	27	por_1	0x1	RW	Illegal NCRD or NCBRD access, Primary Interrupt Enable Bit
EBUS_TO_LOG_EN	26	por_1	0x1	RW	EBus Ready Timeout Detected, Error Log Enable Bit

Table 1-70 JBC Error Log Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
MB_PEA_LOG_EN	25	por_1	0x1	RW	Merge Buffer Address Parity Error Detected, Error Log Enable Bit
MB_PER_LOG_EN	24	por_1	0x1	RW	Merge Buffer Parity Error Detected on read, Error Log Enable Bit
MB_PEW_LOG_EN	23	por_1	0x1	RW	Merge Buffer Parity Error Detected on write, Error Log Enable Bit
UE_ASYNC_LOG_EN	22	por_1	0x1	RW	UE Asynchronous Fault, Error Log Enable Bit
CE_ASYNC_LOG_EN	21	por_1	0x1	RW	CE Asynchronous Fault, Error Log Enable Bit
JTE_LOG_EN	20	por_1	0x1	RW	JBus Time-out Err, Error Log Enable Bit
JBE_LOG_EN	19	por_1	0x1	RW	JBus Bus Error, Error Log Enable Bit
JUE_LOG_EN	18	por_1	0x1	RW	JBus Unmapped Error, Error Log Enable Bit
IJP_LOG_EN	17	por_1	0x1	RW	Invalid JBUS Port Error, Error Log Enable Bit
ICISE_LOG_EN	16	por_1	0x1	RW	JBus Illegal Coherency Install State, Error Log Enable Bit
CPE_LOG_EN	15	por_1	0x1	RW	Control Parity Error, Error Log Enable Bit
APE_LOG_EN	14	por_1	0x1	RW	Address Parity Error, Error Log Enable Bit
WR_DPE_LOG_EN	13	por_1	0x1	RW	Write Data Parity, Error Log Enable Bit
RD_DPE_LOG_EN	12	por_1	0x1	RW	Read Data Parity Error, Error Log Enable Bit.

Table 1-70 JBC Error Log Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
ILL_BMW_LOG_EN	11	por_1	0x1	RW	Illegal byte mask for NCWR, Error Log Enable Bit
ILL_BMR_LOG_EN	10	por_1	0x1	RW	Illegal byte mask for NCRD, Error Log Enable Bit
BJC_LOG_EN	9	por_1	0x1	RW	Bad JBus cmd, Error Log Enable Bit
PIO_UNMAP_LOG_EN	8	por_1	0x1	RW	NCWR or NCBWR Unmapped, Error Log Enable Bit
PIO_DPE_LOG_EN	7	por_1	0x1	RW	PIO Data Parity Error, Error Log Enable Bit
PIO_CPE_LOG_EN	6	por_1	0x1	RW	PIO Command Parity Error, Error Log Enable Bit
ILL_ACC_LOG_EN	5	por_1	0x1	RW	Illegal NCWR or NCBWR access, Primary Interrupt Enable Bit
UNSOL_RD_LOG_EN	4	por_1	0x1	RW	Unsolicited read response, Error Log Enable Bit
UNSOL_INTR_LOG_EN	3	por_1	0x1	RW	Unsolicited interrupt ACK/NAK, Error Log Enable Bit
JTCEEW_LOG_EN	2	por_1	0x1	RW	JBUS Time-out on write error, Error Log Enable Bit
JTCEEI_LOG_EN	1	por_1	0x1	RW	JBUS Time-out on interrupt error, Error Log Enable Bit
JTCEER_LOG_EN	0	por_1	0x1	RW	JBUS Time-out on read error, Error Log Enable Bit

1.3.3.52 JBC Interrupt Enable Register (0x00471008 / 0x0)

This is the JBC Error Log Enable Register. It is used to enable interrupts for all of the possible JBC errors detected by Fire. By default all of the errors are not enabled to generate an interrupt.

Table 1-71 JBC Interrupt Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
SPARE_S_INT_EN	63:61	rst_1	0x0	RW	Spare Errors, Primary Interrupt Enable Bit
PIO_UNMAP_RD_S_INT_EN	60	rst_1	0x0	RW	NCWR or NCBWR Unmapped, Secondary Interrupt Enable Bit
ILL_ACC_RD_S_INT_EN	59	rst_1	0x0	RW	Illegal NCRD or NCBRD access, Secondary Interrupt Enable Bit
EBUS_TO_S_LOG_EN	58	rst_1	0x0	RW	EBus Ready Timeout Detected, Secondary Interrupt Enable Bit
MB_PEA_S_INT_EN	57	rst_1	0x0	RW	Merge Buffer Address Parity Error Detected, Secondary Interrupt Enable Bit
MB_PER_S_INT_EN	56	rst_1	0x0	RW	Merge Buffer Parity Error Detected on read, Secondary Interrupt Enable Bit
MB_PEW_S_INT_EN	55	rst_1	0x0	RW	Merge Buffer Parity Error Detected on write, Secondary Interrupt Enable Bit
UE_ASYNC_S_INT_EN	54	rst_1	0x0	RW	UE Asynchronous Fault, Secondary Interrupt Enable Bit
CE_ASYNC_S_INT_EN	53	rst_1	0x0	RW	CE Asynchronous Fault, Secondary Interrupt Enable Bit
JTE_S_INT_EN	52	rst_1	0x0	RW	JBus Time-out Err, Secondary Interrupt Enable Bit
JBE_S_INT_EN	51	rst_1	0x0	RW	JBus Bus Error, Secondary Interrupt Enable Bit
JUE_S_INT_EN	50	rst_1	0x0	RW	JBus Unmapped Error, Secondary Interrupt Enable Bit

Table 1-71 JBC Interrupt Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
IJP_S_INT_EN	49	rst_1	0x0	RW	Invalid JBUS Port Error, Secondary Interrupt Enable Bit
ICISE_S_INT_EN	48	rst_1	0x0	RW	JBus Illegal Coherency Install State, Secondary Interrupt Enable Bit
CPE_S_INT_EN	47	rst_1	0x0	RW	Control Parity Error, Secondary Interrupt Enable Bit
APE_S_INT_EN	46	rst_1	0x0	RW	Address Parity Error, Secondary Interrupt Enable Bit
WR_DPE_S_INT_EN	45	rst_1	0x0	RW	Write Data Parity, Secondary Interrupt Enable Bit
RD_DPE_S_INT_EN	44	rst_1	0x0	RW	Read Data Parity Error, Secondary Interrupt Enable Bit.
ILL_BMW_S_INT_EN	43	rst_1	0x0	RW	Illegal byte mask for NCWR, Secondary Interrupt Enable Bit
ILL_BMR_S_INT_EN	42	rst_1	0x0	RW	Illegal byte mask for NCRD, Secondary Interrupt Enable Bit
BJC_S_INT_EN	41	rst_1	0x0	RW	Bad JBus cmd, Secondary Interrupt Enable Bit
PIO_UNMAP_S_INT_EN	40	rst_1	0x0	RW	NCWR or NCBWR Unmapped, Secondary Interrupt Enable Bit
PIO_DPE_S_INT_EN	39	rst_1	0x0	RW	PIO Data Parity Error, Secondary Interrupt Enable Bit
PIO_CPE_S_INT_EN	38	rst_1	0x0	RW	PIO Command Parity Error, Secondary Interrupt Enable Bit
ILL_ACC_S_INT_EN	37	rst_1	0x0	RW	Illegal NCWR or NCBWR access, Secondary Interrupt Enable Bit
UNSOL_RD_S_INT_EN	36	rst_1	0x0	RW	Unsolicited read response, Secondary Interrupt Enable Bit
UNSOL_INTR_S_INT_EN	35	rst_1	0x0	RW	Unsolicited interrupt ACK/NAK, Secondary Interrupt Enable Bit

Table 1-71 JBC Interrupt Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
JTCEEW_S_INT_EN	34	rst_1	0x0	RW	JBus Time-out on write error, Secondary Interrupt Enable Bit
JTCEEI_S_INT_EN	33	rst_1	0x0	RW	JBus Time-out on interrupt error, Secondary Interrupt Enable Bit
JTCEER_S_INT_EN	32	rst_1	0x0	RW	JBus Time-out on read error, Secondary Interrupt Enable Bit
SPARE_P_INT_EN	31:29	rst_1	0x0	RW	Spare Errors, Primary Interrupt Enable Bit
PIO_UNMAP_RD_P_INT_EN	28	rst_1	0x0	RW	NCWR or NCBWR Unmapped, Primary Interrupt Enable Bit
ILL_ACC_RD_P_INT_EN	27	rst_1	0x0	RW	Illegal NCRD or NCBRD access, Primary Interrupt Enable Bit
EBUS_TO_P_LOG_EN	26	rst_1	0x0	RW	EBus Ready Timeout Detected, Primary Interrupt Enable Bit
MB_PEA_P_INT_EN	25	rst_1	0x0	RW	Merge Buffer Address Parity Error Detected, Primary Interrupt Enable Bit
MB_PER_P_INT_EN	24	rst_1	0x0	RW	Merge Buffer Parity Error Detected on read, Primary Interrupt Enable Bit
MB_PEW_P_INT_EN	23	rst_1	0x0	RW	Merge Buffer Parity Error Detected on write, Primary Interrupt Enable Bit
UE_ASYNC_P_INT_EN	22	rst_1	0x0	RW	UE Asynchronous Fault, Primary Interrupt Enable Bit
CE_ASYNC_P_INT_EN	21	rst_1	0x0	RW	CE Asynchronous Fault, Primary Interrupt Enable Bit
JTE_P_INT_EN	20	rst_1	0x0	RW	JBus Time-out Err, Primary Interrupt Enable Bit
JBE_P_INT_EN	19	rst_1	0x0	RW	JBus Bus Error, Primary Interrupt Enable Bit

Table 1-71 JBC Interrupt Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
JUE_P_INT_EN	18	rst_1	0x0	RW	JBus Unmapped Error, Primary Interrupt Enable Bit
IJP_P_INT_EN	17	rst_1	0x0	RW	Invalid JBUS Port Error, Primary Interrupt Enable Bit
ICISE_P_INT_EN	16	rst_1	0x0	RW	JBus Illegal Coherency Install State, Primary Interrupt Enable Bit
CPE_P_INT_EN	15	rst_1	0x0	RW	Control Parity Error, Primary Interrupt Enable Bit
APE_P_INT_EN	14	rst_1	0x0	RW	Address Parity Error, Primary Interrupt Enable Bit
WR_DPE_P_INT_EN	13	rst_1	0x0	RW	Write Data Parity, Primary Interrupt Enable Bit
RD_DPE_P_INT_EN	12	rst_1	0x0	RW	Read Data Parity Error, Primary Interrupt Enable Bit.
ILL_BMW_P_INT_EN	11	rst_1	0x0	RW	Illegal byte mask for NCWR, Primary Interrupt Enable Bit
ILL_BMR_P_INT_EN	10	rst_1	0x0	RW	Illegal byte mask for NCRD, Primary Interrupt Enable Bit
BJC_P_INT_EN	9	rst_1	0x0	RW	Bad JBus cmd, Primary Interrupt Enable Bit
PIO_UNMAP_P_INT_EN	8	rst_1	0x0	RW	NCWR or NCBWR Unmapped, Primary Interrupt Enable Bit
PIO_DPE_P_INT_EN	7	rst_1	0x0	RW	PIO Data Parity Error, Primary Interrupt Enable Bit
PIO_CPE_P_INT_EN	6	rst_1	0x0	RW	PIO Command Parity Error, Primary Interrupt Enable Bit
ILL_ACC_P_INT_EN	5	rst_1	0x0	RW	Illegal NCWR or NCBWR access, Primary Interrupt Enable Bit
UNSOL_RD_P_INT_EN	4	rst_1	0x0	RW	Unsolicited read response, Primary Interrupt Enable Bit

Table 1-71 JBC Interrupt Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
UNSOL_INTR_P_INT_EN	3	rst_1	0x0	RW	Unsolicited interrupt ACK/NAK, Primary Interrupt Enable Bit
JTCEEW_P_INT_EN	2	rst_1	0x0	RW	JBus Time-out on write error, Primary Interrupt Enable Bit
JTCEEI_P_INT_EN	1	rst_1	0x0	RW	JBus Time-out on interrupt error, Primary Interrupt Enable Bit
JTCEER_P_INT_EN	0	rst_1	0x0	RW	JBus Time-out on read error, Primary Interrupt Enable Bit

1.3.3.53 JBC Interrupt Status Register (0x00471010/0x0)

This is the JBC Interrupt Status Register. This is a read only register which SW reads to obtain the source of interrupt for the JBC core. (Mondo number 63). This register will contain a 1 in all bits where an interrupt was generated for.

Table 1-72 JBC Interrupt Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
SPARE_S	63:61	rst_1	0x0	R	Spare Errors, Secondary Error Status Bit; 1 = Error Received. Should NEVER be set.
PIO_UNMAP_RD_S	60	rst_1	0x0	R	NCRD or NCBRD Unmapped, A NCRD or NCBRD transaction did not map to any valid address space enabled by Fire. Secondary Error Status Bit; 1 = Error Received
ILL_ACC_RD_S	59	rst_1	0x0	R	Illegal NCRD or NCBRD Access: Config or IO access of wrong size (e.g. 8 bytes) I2C illegal size EBus Illegal size NCBRD to CSR space CSR Illegal Size. Secondary Error Status Bit; 1 = Error Received

Table 1-72 JBC Interrupt Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
EBUS_TO_S	58	rst_1	0x0	R	EBus Ready Timeout Error Detected. Secondary Error Status Bit; 1 = Error Received
MB_PEA_S	57	rst_1	0x0	R	Merge Buffer Address Parity Error Detected. FATAL error. Secondary Error Status Bit; 1 = Error Received
MB_PER_S	56	rst_1	0x0	R	Merge Buffer Parity Error Detected on read. Secondary Error Status Bit; 1 = Error Received
MB_PEW_S	55	rst_1	0x0	R	Merge Buffer Parity Error Detected on write. Secondary Error Status Bit; 1 = Error Received
UE_ASYNC_S	54	rst_1	0x0	R	UE Asynchronous Fault. Secondary Error Status Bit; 1 = Error Received
CE_ASYNC_S	53	rst_1	0x0	R	CE Asynchronous Fault. Secondary Error Status Bit; 1 = Error Received
JTE_S	52	rst_1	0x0	R	JBus Time-out Err. Fire received a JBus data packet for a cacheable DMA read with Read Error indicating a Time-out. Secondary Error Status Bit; 1 = Error Received
JBE_S	51	rst_1	0x0	R	JBus Bus Error. Fire received a JBus data packet for a DMA read with Read Error indicating a Bus Error. Secondary Error Status Bit; 1 = Error Received

Table 1-72 JBC Interrupt Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
JUE_S	50	rst_1	0x0	R	<p>JBus Unmapped Error. This bit gets set when any one of the following conditions is true:</p> <ul style="list-style-type: none"> - Fire received a JBus data packet for a DMA read with Read Error indicating an Unmapped Error. - Fire received a write Packet with an address beyond its range. - Fire received a non 8 byte aligned NCWR for a TSB Flush - Fire received NCBWR to 8 MB address space <p>Secondary Error Status Bit; 1 = Error Received</p>
IJP_S	49	rst_1	0x0	R	<p>Invalid JBUS Port Error. This bit gets set when Fire issues a Write to an invalid JBus port. Secondary Error Status Bit; 1 = Error Received</p>
ICISE_S	48	rst_1	0x0	R	<p>JBus Illegal Coherency Install State Error. Illegal data install state of 'O'. Secondary Error Status Bit; 1 = Error Received</p>
CPE_S	47	rst_1	0x0	R	<p>Control Parity Error. FATAL error. Control parity error detected by Fire on the JBus J_PAR line. Only valid if Fire Control and Status Register bit 43 = 1. Secondary Error Status Bit; 1 = Error Received</p>
APE_S	46	rst_1	0x0	R	<p>Address Parity Error. FATAL error. Address parity error detected by Fire on J_AD bus. Secondary Error Status Bit; 1 = Error Received</p>

Table 1-72 JBC Interrupt Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
WR_DPE_S	45	rst_1	0x0	R	Write Data Parity Error. Write Data Cycle Parity error detected by Fire on J_AD bus. Secondary Error Status Bit; 1 = Error Received
RD_DPE_S	44	rst_1	0x0	R	Read Data Parity Error. Read Data Cycle Parity error detected by Fire on J_AD bus. Secondary Error Status Bit; 1 = Error Received
ILL_BMW_S	43	rst_1	0x0	R	Illegal byte mask detected for NCWR. Secondary Error Status Bit; 1 = Error Received
ILL_BMR_S	42	rst_1	0x0	R	Illegal byte mask detected for NCRD. Secondary Error Status Bit; 1 = Error Received
BJC_S	41	rst_1	0x0	R	Bad JBus cmd. Unrecognized JBus command received. Secondary Error Status Bit; 1 = Error Received
PIO_UNMAP_S	40	rst_1	0x0	R	NCWR or NCBWR Unmapped. A NCWR or NCBWR transaction did not map to any valid address space enabled by Fire. Secondary Error Status Bit; 1 = Error Received
PIO_DPE_S	39	rst_1	0x0	R	PIO Data Parity Error. Fire detected a Parity Error reading out of the PIO Data Ram in the dmcint block on a NCWR or NCBWR command. Secondary Error Status Bit; 1 = Error Received

Table 1-72 JBC Interrupt Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
PIO_CPE_S	38	rst_1	0x0	R	PIO Command Parity Error. Fire detected a Parity Error reading out of the PIO Data Ram in the dmcint block on a NCWR, NCBWR, NCRD, or NCBRD command. Secondary Error Status Bit; 1 = Error Received
ILL_ACC_S	37	rst_1	0x0	R	Illegal NCWR or NCBWR Access: Config or IO access of wrong size (e.g. 8 bytes) I2C illegal size EBus Illegal size NCBWR to CSR space CSR Illegal Size. Secondary Error Status Bit; 1 = Error Received
UNSOL_RD_S	36	rst_1	0x0	R	Unsolicited read response received. Secondary Error Status Bit; 1 = Error Received
UNSOL_INTR_S	35	rst_1	0x0	R	Unsolicited interrupt ACK/NAK received. Secondary Error Status Bit; 1 = Error Received
JTCEEW_S	34	rst_1	0x0	R	JBus Time-out Counter Expired on write error. FATAL error. Fire detected a Time-out for a write because the JBus watchdog timer time-out interval was exceeded. Secondary Error Status Bit; 1 = Error Received
JTCEEI_S	33	rst_1	0x0	R	JBus Time-out Counter Expired on interrupt error. FATAL error. Fire detected a Time-out for an interrupt because the JBus watchdog timer time-out interval was exceeded. Secondary Error Status Bit; 1 = Error Received

Table 1-72 JBC Interrupt Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
JTCEER_S	32	rst_1	0x0	R	JBus Time-out Counter Expired on read error. FATAL error. Fire detected a Time-out for a read because the JBus watchdog timer time-out interval was exceeded. Secondary Error Status Bit; 1 = Error Received
SPARE_P	31:29	rst_1	0x0	R	Spare Errors, Primary Error Status Bit; 1 = Error Received. Should NEVER be set
PIO_UNMAP_RD_P	28	rst_1	0x0	R	NCRD or NCBRD Unmapped. A NCRD or NCBRD transaction did not map to any valid address space enabled by Fire. Primary Error Status Bit; 1 = Error Received
ILL_ACC_RD_P	27	rst_1	0x0	R	Illegal NCRD or NCBRD Access: Config or IO access of wrong size (e.g. 8 bytes) I2C illegal size EBus Illegal size CSR Illegal Size. Primary Error Status Bit; 1 = Error Received
EBUS_TO_P	26	rst_1	0x0	R	EBus Ready Timeout Error Detected. Primary Error Status Bit; 1 = Error Received
MB_PEA_P	25	rst_1	0x0	R	Merge Buffer Address Parity Error Detected. FATAL error. Primary Error Status Bit; 1 = Error Received
MB_PER_P	24	rst_1	0x0	R	Merge Buffer Parity Error Detected on read. Primary Error Status Bit; 1 = Error Received
MB_PEW_P	23	rst_1	0x0	R	Merge Buffer Parity Error Detected on write. Primary Error Status Bit; 1 = Error Received

Table 1-72 JBC Interrupt Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
UE_ASYNC_P	22	rst_1	0x0	R	UE Asynchronous Fault. Primary Error Status Bit; 1 = Error Received
CE_ASYNC_P	21	rst_1	0x0	R	CE Asynchronous Fault. Primary Error Status Bit; 1 = Error Received
JTE_P	20	rst_1	0x0	R	JBus Time-out Err. Fire received a JBus data packet for a noncacheable DMA read with Read Error indicating a Time-out. Primary Error Status Bit; 1 = Error Received
JBE_P	19	rst_1	0x0	R	JBus Bus Error. Fire received a JBus data packet for a noncacheable DMA read with Read Error indicating a Bus Error. Primary Error Status Bit; 1 = Error Received
JUE_P	18	rst_1	0x0	R	JBus Unmapped Error. This bit gets set when any one of the following conditions is true: <ul style="list-style-type: none"> - Fire received a JBus data packet for a DMA read with Read Error indicating an Unmapped Error. - Fire received a write Packet with an address beyond its range. - Fire received a non 8 byte aligned NCWR for a TSB flush - Fire received NCBWR to 8 MB address space Primary Error Status Bit; 1 = Error Received
IJP_P	17	rst_1	0x0	R	Invalid JBUS Port Error. This bit gets set when Fire issues a Write to an invalid JBus port. Primary Error Status Bit; 1 = Error Received

Table 1-72 JBC Interrupt Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
ICISE_P	16	rst_1	0x0	R	JBus Illegal Coherency Install State Error. Illegal data install state of 'O'. Primary Error Status Bit; 1 = Error Received
CPE_P	15	rst_1	0x0	R	Control Parity Error. FATAL error. Control parity error detected by Fire on the JBus J_PAR line. Only valid if Fire Control and Status Register bit 43 = 1. Primary Error Status Bit; 1 = Error Received
APE_P	14	rst_1	0x0	R	Address Parity Error. FATAL error. Address parity error detected by Fire on J_AD bus. Primary Error Status Bit; 1 = Error Received
WR_DPE_P	13	rst_1	0x0	R	Write Data Parity Error. Write Data Cycle Parity or UE (j_adtype[4] == 1) error detected by Fire on J_AD bus. Primary Error Status Bit; 1 = Error Received
RD_DPE_P	12	rst_1	0x0	R	Read Data Parity Error. Read Data Cycle Parity error detected by Fire on J_AD bus. Primary Error Status Bit; 1 = Error Received
ILL_BMW_P	11	rst_1	0x0	R	Illegal byte mask detected for NCWR. Primary Error Status Bit; 1 = Error Received
ILL_BMR_P	10	rst_1	0x0	R	Illegal byte mask detected for NCRD. Primary Error Status Bit; 1 = Error Received
BJC_P	9	rst_1	0x0	R	Bad JBus cmd. Unrecognized JBus command received. Primary Error Status Bit; 1 = Error Received

Table 1-72 JBC Interrupt Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
PIO_UNMAP_P	8	rst_1	0x0	R	NCWR or NCBWR Unmapped. A NCWR or NCBWR transaction did not map to any valid address space enabled by Fire. Primary Error Status Bit; 1 = Error Received
PIO_DPE_P	7	rst_1	0x0	R	PIO Data Parity Error. Fire detected a Parity Error reading out of the PIO Data Ram in the dmcint block on a NCWR or NCBWR command. Primary Error Status Bit; 1 = Error Received
PIO_CPE_P	6	rst_1	0x0	R	PIO Command Parity Error. Fire detected a Parity Error reading out of the PIO Data Ram in the dmcint block on a NCWR, NCBWR, NCRD, or NCBRD command. Primary Error Status Bit; 1 = Error Received
ILL_ACC_P	5	rst_1	0x0	R	Illegal NCWR or NCBWR Access: Config or IO access of wrong size (e.g. 8 bytes) I2C illegal size EBus Illegal size CSR Illegal Size. Primary Error Status Bit; 1 = Error Received
UNSOL_RD_P	4	rst_1	0x0	R	Unsolicited read response received. Primary Error Status Bit; 1 = Error Received
UNSOL_INTR_P	3	rst_1	0x0	R	Unsolicited interrupt ACK/NAK received. Primary Error Status Bit; 1 = Error Received

Table 1-72 JBC Interrupt Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
JTCEEW_P	2	rst_1	0x0	R	JBus Time-out Counter Expired on write error. FATAL error. Fire detected a Time-out for a write because the JBus watchdog timer time-out interval was exceeded. Primary Error Status Bit; 1 = Error Received
JTCEEL_P	1	rst_1	0x0	R	JBus Time-out Counter Expired on interrupt error. FATAL error. Fire detected a Time-out for an interrupt because the JBus watchdog timer time-out interval was exceeded. Primary Error Status Bit; 1 = Error Received
JTCEER_P	0	rst_1	0x0	R	JBus Time-out Counter Expired on read error. FATAL error. Fire detected a Time-out for a read because the JBus watchdog timer time-out interval was exceeded. Primary Error Status Bit; 1 = Error Received

1.3.3.54 JBC Error Status Clear Register (0x00471018 / 0x0)

This is the JBC Error Status Clear Register. This is a read, write one to clear register which serves two purposes for SW. First, SW can read this register to determine which JBC errors have been logged by Fire but have not necessarily caused an interrupt (Mondo 63). This register will contain a 1 in all bits where errors were logged. Second, SW uses this register to clear the JBC errors detected by Fire. It writes a 1 to the bit(s) which it wants to clear.

Table 1-73 JBC Error Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
SPARE_S	63:61	por_1	0x0	RW1C	Spare Errors, Secondary Error Status Bit; 1 = Error Received. Should NEVER be set

Table 1-73 JBC Error Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
PIO_UNMAP_RD_S	60	por_1	0x0	RW1C	NCRD or NCBRD Unmapped, A NCRD or NCBRD transaction did not map to any valid address space enabled by Fire. Secondary Error Status Bit; 1 = Error Received
ILL_ACC_RD_S	59	por_1	0x0	RW1C	Illegal NCRD or NCBRD Access: Config or IO access of wrong size (e.g. 8 bytes) I2C illegal size EBus Illegal size CSR Illegal Size. Secondary Error Status Bit; 1 = Error Received
EBUS_TO_S	58	por_1	0x0	RW1C	EBus Ready Timeout Error Detected. Secondary Error Status Bit; 1 = Error Received
MB_PEA_S	57	por_1	0x0	RW1C	Merge Buffer Address Parity Error Detected. FATAL error. Secondary Error Status Bit; 1 = Error Received
MB_PER_S	56	por_1	0x0	RW1C	Merge Buffer Parity Error Detected on read. Secondary Error Status Bit; 1 = Error Received
MB_PEW_S	55	por_1	0x0	RW1C	Merge Buffer Parity Error Detected on write. Secondary Error Status Bit; 1 = Error Received
UE_ASYNC_S	54	por_1	0x0	RW1C	UE Asynchronous Fault. Secondary Error Status Bit; 1 = Error Received
CE_ASYNC_S	53	por_1	0x0	RW1C	CE Asynchronous Fault. Secondary Error Status Bit; 1 = Error Received

Table 1-73 JBC Error Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
JTE_S	52	por_1	0x0	RW1C	JBus Time-out Err. Fire received a JBus data packet for a noncacheable DMA read with Read Error indicating a Time-out. Secondary Error Status Bit; 1 = Error Received
JBE_S	51	por_1	0x0	RW1C	JBus Bus Error. Fire received a JBus data packet for a DMA read with Read Error indicating a Bus Error. Secondary Error Status Bit; 1 = Error Received
JUE_S	50	por_1	0x0	RW1C	JBus Unmapped Error. This bit gets set when any one of the following conditions is true: <ul style="list-style-type: none"> - Fire received a JBus data packet for a DMA read with Read Error indicating an Unmapped Error. - Fire received a write Packet with an address beyond its range. - Fire received a non 8 byte aligned NCWR for a TSB flush - Fire received NCBWR to 8 MB address space Secondary Error Status Bit; 1 = Error Received
IJP_S	49	por_1	0x0	RW1C	Invalid JBUS Port Error. This bit gets set when Fire issues a Write to an invalid JBus port. Secondary Error Status Bit; 1 = Error Received
ICISE_S	48	por_1	0x0	RW1C	JBus Illegal Coherency Install State Error. Illegal data install state of 'O'. Secondary Error Status Bit; 1 = Error Received

Table 1-73 JBC Error Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
CPE_S	47	por_1	0x0	RW1C	Control Parity Error. FATAL error. Control parity error detected by Fire on the JBus J_PAR line. Only valid if Fire Control and Status Register bit 43 = 1. Secondary Error Status Bit; 1 = Error Received
APE_S	46	por_1	0x0	RW1C	Address Parity Error. FATAL error. Address parity error detected by Fire on J_AD bus. Secondary Error Status Bit; 1 = Error Received
WR_DPE_S	45	por_1	0x0	RW1C	Write Data Parity Error. Write Data Cycle Parity error detected by Fire on J_AD bus. Secondary Error Status Bit; 1 = Error Received
RD_DPE_S	44	por_1	0x0	RW1C	Read Data Parity Error. Read Data Cycle Parity error detected by Fire on J_AD bus. Secondary Error Status Bit; 1 = Error Received
ILL_BMW_S	43	por_1	0x0	RW1C	Illegal byte mask detected for NCWR. Secondary Error Status Bit; 1 = Error Received
ILL_BMR_S	42	por_1	0x0	RW1C	Illegal byte mask detected for NCRD. Secondary Error Status Bit; 1 = Error Received
BJC_S	41	por_1	0x0	RW1C	Bad JBus cmd. Unrecognized JBus command received. Secondary Error Status Bit; 1 = Error Received

Table 1-73 JBC Error Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
PIO_UNMAP_S	40	por_1	0x0	RW1C	NCWR or NCBWR Unmapped, A NCWR or NCBWR transaction did not map to any valid address space enabled by Fire. Secondary Error Status Bit; 1 = Error Received
PIO_DPE_S	39	por_1	0x0	RW1C	PIO Data Parity Error, Fire detected a Parity Error reading out of the PIO Data Ram in the dmcint block on a NCWR or NCBWR command. Secondary Error Status Bit; 1 = Error Received
PIO_CPE_S	38	por_1	0x0	RW1C	PIO Command Parity Error, Fire detected a Parity Error reading out of the PIO Data Ram in the dmcint block on a NCWR, NCBWR, NCRD, or NCBRD command. Secondary Error Status Bit; 1 = Error Received
ILL_ACC_S	37	por_1	0x0	RW1C	Illegal NCWR or NCBWR Access: Config or IO access of wrong size (e.g. 8 bytes) I2C illegal size EBus Illegal size CSR Illegal Size. Secondary Error Status Bit; 1 = Error Received
UNSOL_RD_S	36	por_1	0x0	RW1C	Unsolicited read response received. Secondary Error Status Bit; 1 = Error Received
UNSOL_INTR_S	35	por_1	0x0	RW1C	Unsolicited interrupt ACK/NAK received. Secondary Error Status Bit; 1 = Error Received

Table 1-73 JBC Error Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
JTCEEW_S	34	por_1	0x0	RW1C	JBus Time-out Counter Expired on write error. FATAL error. Fire detected a Time-out for a write because the JBus watchdog timer time-out interval was exceeded. Secondary Error Status Bit; 1 = Error Received
JTCEEL_S	33	por_1	0x0	RW1C	JBus Time-out Counter Expired on interrupt error. FATAL error. Fire detected a Time-out for an interrupt because the JBus watchdog timer time-out interval was exceeded. Secondary Error Status Bit; 1 = Error Received
JTCEER_S	32	por_1	0x0	RW1C	JBus Time-out Counter Expired on read error. FATAL error. Fire detected a Time-out for a read because the JBus watchdog timer time-out interval was exceeded. Secondary Error Status Bit; 1 = Error Received
SPARE_P	31:29	por_1	0x0	RW1C	Spare Errors, Primary Error Status Bit; 1 = Error Received. Should NEVER be set
PIO_UNMAP_RD_P	28	por_1	0x0	RW1C	NCRD or NCBRD Unmapped, A NCRD or NCBRD transaction did not map to any valid address space enabled by Fire. Primary Error Status Bit; 1 = Error Received
ILL_ACC_RD_P	27	por_1	0x0	RW1C	Illegal NCRD or NCBRD Access: Config or IO access of wrong size (e.g. 8 bytes) I2C illegal size EBus Illegal size CSR Illegal Size. Primary Error Status Bit; 1 = Error Received

Table 1-73 JBC Error Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
EBUS_TO_P	26	por_1	0x0	RW1C	EBus Ready Timeout Error Detected. Primary Error Status Bit; 1 = Error Received
MB_PEA_P	25	por_1	0x0	RW1C	Merge Buffer Address Parity Error Detected. FATAL error. Primary Error Status Bit; 1 = Error Received
MB_PER_P	24	por_1	0x0	RW1C	Merge Buffer Parity Error Detected on read. Primary Error Status Bit; 1 = Error Received
MB_PEW_P	23	por_1	0x0	RW1C	Merge Buffer Parity Error Detected on write. Primary Error Status Bit; 1 = Error Received
UE_ASYN_P	22	por_1	0x0	RW1C	UE Asynchronous Fault. Primary Error Status Bit; 1 = Error Received
CE_ASYN_P	21	por_1	0x0	RW1C	CE Asynchronous Fault. Primary Error Status Bit; 1 = Error Received
JTE_P	20	por_1	0x0	RW1C	JBus Time-out Err. Fire received a JBus data packet for a noncacheable DMA read with Read Error indicating a Time-out. Primary Error Status Bit; 1 = Error Received
JBE_P	19	por_1	0x0	RW1C	JBus Bus Error. Fire received a JBus data packet for a noncacheable DMA read with Read Error indicating a Bus Error. Primary Error Status Bit; 1 = Error Received

Table 1-73 JBC Error Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
JUE_P	18	por_1	0x0	RW1C	JBus Unmapped Error. This bit gets set when any one of the following conditions is true: <ul style="list-style-type: none"> - Fire received a JBus data packet for a DMA read with Read Error indicating an Unmapped Error. - Fire received a write Packet with an address beyond its range. - Fire received a non 8 byte aligned NCWR for a TSB flush - Fire received NCBWR to 8 MB address space Primary Error Status Bit; 1 = Error Received
IJP_P	17	por_1	0x0	RW1C	Invalid JBUS Port Error. This bit gets set when Fire issues a Write to an invalid JBus port. Primary Error Status Bit; 1 = Error Received
ICISE_P	16	por_1	0x0	RW1C	JBus Illegal Coherency Install State Error. Illegal data install state of 'O'. Primary Error Status Bit; 1 = Error Received
CPE_P	15	por_1	0x0	RW1C	Control Parity Error. FATAL error. Control parity error detected by Fire on the JBus J_PAR line. Only valid if Fire Control and Status Register bit 43 = 1. Primary Error Status Bit; 1 = Error Received
APE_P	14	por_1	0x0	RW1C	Address Parity Error. FATAL error. Address parity error detected by Fire on J_AD bus. Primary Error Status Bit; 1 = Error Received
WR_DPE_P	13	por_1	0x0	RW1C	Write Data Parity Error. Write Data Cycle Parity error detected by Fire on J_AD bus. Primary Error Status Bit; 1 = Error Received

Table 1-73 JBC Error Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
RD_DPE_P	12	por_1	0x0	RW1C	Read Data Parity Error. Read Data Cycle Parity error detected by Fire on J_AD bus. Primary Error Status Bit; 1 = Error Received
ILL_BMW_P	11	por_1	0x0	RW1C	Illegal byte mask detected for NCWR. Primary Error Status Bit; 1 = Error Received
ILL_BMR_P	10	por_1	0x0	RW1C	Illegal byte mask detected for NCRD. Primary Error Status Bit; 1 = Error Received
BJC_P	9	por_1	0x0	RW1C	Bad JBus cmd. Unrecognized JBus command received. Primary Error Status Bit; 1 = Error Received
PIO_UNMAP_P	8	por_1	0x0	RW1C	NCWR or NCBWR Unmapped, A NCWR or NCBWR transaction did not map to any valid address space enabled by Fire. Primary Error Status Bit; 1 = Error Received
PIO_DPE_P	7	por_1	0x0	RW1C	PIO Data Parity Error, Fire detected a Parity Error reading out of the PIO Data Ram in the dmcint block on a NCWR or NCBWR command. Primary Error Status Bit; 1 = Error Received
PIO_CPE_P	6	por_1	0x0	RW1C	PIO Command Parity Error, Fire detected a Parity Error reading out of the PIO Data Ram in the dmcint block on a NCWR, NCBWR, NCRD, or NCBRD command. Primary Error Status Bit; 1 = Error Received

Table 1-73 JBC Error Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
ILL_ACC_P	5	por_1	0x0	RW1C	Illegal NCWR or NCBWR Access: Config or IO access of wrong size (e.g. 8 bytes) I2C illegal size EBus Illegal size CSR Illegal Size. Primary Error Status Bit; 1 = Error Received
UNSOL_RD_P	4	por_1	0x0	RW1C	Unsolicited read response received. Primary Error Status Bit; 1 = Error Received
UNSOL_INTR_P	3	por_1	0x0	RW1C	Unsolicited interrupt ACK/NAK received. Primary Error Status Bit; 1 = Error Received
JTCEEW_P	2	por_1	0x0	RW1C	JBus Time-out Counter Expired on write error. FATAL error. Fire detected a Time-out for a write because the JBus watchdog timer time-out interval was exceeded. Primary Error Status Bit; 1 = Error Received
JTCEEI_P	1	por_1	0x0	RW1C	JBus Time-out Counter Expired on interrupt error. FATAL error. Fire detected a Time-out for an interrupt because the JBus watchdog timer time-out interval was exceeded. Primary Error Status Bit; 1 = Error Received
JTCEER_P	0	por_1	0x0	RW1C	JBus Time-out Counter Expired on read error. FATAL error. Fire detected a Time-out for a read because the JBus watchdog timer time-out interval was exceeded. Primary Error Status Bit; 1 = Error Received

1.3.3.55 JBC Error Status Set Register (0x00471020/0x0)

This is the JBC Error Status Set Register. This is a read, write one to set register which serves two purposes for SW. First, SW can read this register to determine which JBC errors have been logged by Fire but have not necessarily caused an interrupt (Mondo 63). This register will contain a 1 in all bits where errors were logged. Second, SW uses this register to set the JBC errors detected by Fire. It writes a 1 to the bit(s) which it wants to set. THIS SHOULD ONLY BE USED FOR DIAG PURPOSES AND ERROR TESTING.

Table 1-74 JBC Error Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
SPARE_S	63:61	por_1	0x0	RW1S	Spare Errors, Secondary Error Status Bit; 1 = Error Received. Should NEVER be set
PIO_UNMAP_RD_S	60	por_1	0x0	RW1S	NCRD or NCBRD Unmapped, A NCRD or NCBRD transaction did not map to any valid address space enabled by Fire. Secondary Error Status Bit; 1 = Error Received
ILL_ACC_RD_S	59	por_1	0x0	RW1S	Illegal NCRD or NCBRD Access: Config or IO access of wrong size (e.g. 8 bytes) I2C illegal size EBus Illegal size CSR Illegal Size. Secondary Error Status Bit; 1 = Error Received
EBUS_TO_S	58	por_1	0x0	RW1S	EBus Ready Timeout Error Detected. Secondary Error Status Bit; 1 = Error Received
MB_PEA_S	57	por_1	0x0	RW1S	Merge Buffer Address Parity Error Detected. FATAL error. Secondary Error Status Bit; 1 = Error Received
MB_PER_S	56	por_1	0x0	RW1S	Merge Buffer Parity Error Detected on read. Secondary Error Status Bit; 1 = Error Received

Table 1-74 JBC Error Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
MB_PEW_S	55	por_1	0x0	RW1S	Merge Buffer Parity Error Detected on write. Secondary Error Status Bit; 1 = Error Received
UE_ASYN_S	54	por_1	0x0	RW1S	UE Asynchronous Fault. Secondary Error Status Bit; 1 = Error Received
CE_ASYN_S	53	por_1	0x0	RW1S	CE Asynchronous Fault. Secondary Error Status Bit; 1 = Error Received
JTE_S	52	por_1	0x0	RW1S	JBus Time-out Err. Fire received a JBus data packet for a noncacheable DMA read with Read Error indicating a Time-out. Secondary Error Status Bit; 1 = Error Received
JBE_S	51	por_1	0x0	RW1S	JBus Bus Error. Fire received a JBus data packet for a DMA read with Read Error indicating a Bus Error. Secondary Error Status Bit; 1 = Error Received
JUE_S	50	por_1	0x0	RW1S	JBus Unmapped Error. This bit gets set when any one of the following conditions is true: <ul style="list-style-type: none"> - Fire received a JBus data packet for a DMA read with Read Error indicating an Unmapped Error. - Fire received a write Packet with an address beyond its range. - Fire received a non 8 byte aligned NCWR for a TSB flush - Fire received NCBWR to 8 MB address space Secondary Error Status Bit; 1 = Error Received

Table 1-74 JBC Error Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
IJP_S	49	por_1	0x0	RW1S	Invalid JBUS Port Error. This bit gets set when Fire issues a Write to an invalid JBus port. Secondary Error Status Bit; 1 = Error Received
ICISE_S	48	por_1	0x0	RW1S	JBUS Illegal Coherency Install State Error. Illegal data install state of 'O'. Secondary Error Status Bit; 1 = Error Received
CPE_S	47	por_1	0x0	RW1S	Control Parity Error. FATAL error. Control parity error detected by Fire on the JBus J_PAR line. Only valid if Fire Control and Status Register bit 43 = 1. Secondary Error Status Bit; 1 = Error Received
APE_S	46	por_1	0x0	RW1S	Address Parity Error. FATAL error. Address parity error detected by Fire on J_AD bus. Secondary Error Status Bit; 1 = Error Received
WR_DPE_S	45	por_1	0x0	RW1S	Write Data Parity Error. Write Data Cycle Parity error detected by Fire on J_AD bus. Secondary Error Status Bit; 1 = Error Received
RD_DPE_S	44	por_1	0x0	RW1S	Read Data Parity Error. Read Data Cycle Parity error detected by Fire on J_AD bus. Secondary Error Status Bit; 1 = Error Received
ILL_BMW_S	43	por_1	0x0	RW1S	Illegal byte mask detected for NCWR. Secondary Error Status Bit; 1 = Error Received
ILL_BMR_S	42	por_1	0x0	RW1S	Illegal byte mask detected for NCRD. Secondary Error Status Bit; 1 = Error Received

Table 1-74 JBC Error Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
BJC_S	41	por_1	0x0	RW1S	Bad JBus cmd. Unrecognized JBus command received. Secondary Error Status Bit; 1 = Error Received
PIO_UNMAP_S	40	por_1	0x0	RW1S	NCWR or NCBWR Unmapped, A NCWR or NCBWR transaction did not map to any valid address space enabled by Fire. Secondary Error Status Bit; 1 = Error Received
PIO_DPE_S	39	por_1	0x0	RW1S	PIO Data Parity Error, Fire detected a Parity Error reading out of the PIO Data Ram in the dmcint block on a NCWR or NCBWR command. Secondary Error Status Bit; 1 = Error Received
PIO_CPE_S	38	por_1	0x0	RW1S	PIO Command Parity Error, Fire detected a Parity Error reading out of the PIO Data Ram in the dmcint block on a NCWR, NCBWR, NCRD, or NCBRD command. Secondary Error Status Bit; 1 = Error Received
ILL_ACC_S	37	por_1	0x0	RW1S	Illegal NCWR or NCBWR Access: Config or IO access of wrong size (e.g. 8 bytes) I2C illegal size EBus Illegal size CSR Illegal Size. Secondary Error Status Bit; 1 = Error Received
UNSOL_RD_S	36	por_1	0x0	RW1S	Unsolicited read response received. Secondary Error Status Bit; 1 = Error Received

Table 1-74 JBC Error Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
UNSOL_INTR_S	35	por_1	0x0	RW1S	Unsolicited interrupt ACK/NAK received. Secondary Error Status Bit; 1 = Error Received
JTCEEW_S	34	por_1	0x0	RW1S	JBus Time-out Counter Expired on write error. FATAL error. Fire detected a Time-out for a write because the JBus watchdog timer time-out interval was exceeded. Secondary Error Status Bit; 1 = Error Received
JTCEEI_S	33	por_1	0x0	RW1S	JBus Time-out Counter Expired on interrupt error. FATAL error. Fire detected a Time-out for an interrupt because the JBus watchdog timer time-out interval was exceeded. Secondary Error Status Bit; 1 = Error Received
JTCEER_S	32	por_1	0x0	RW1S	JBus Time-out Counter Expired on read error. FATAL error. Fire detected a Time-out for a read because the JBus watchdog timer time-out interval was exceeded. Secondary Error Status Bit; 1 = Error Received
SPARE_P	31:29	por_1	0x0	RW1S	Spare Errors, Primary Error Status Bit; 1 = Error Received. Should NEVER be set
PIO_UNMAP_RD_P	28	por_1	0x0	RW1S	NCRD or NCBRD Unmapped, A NCRD or NCBRD transaction did not map to any valid address space enabled by Fire. Primary Error Status Bit; 1 = Error Received

Table 1-74 JBC Error Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
ILL_ACC_RD_P	27	por_1	0x0	RW1S	Illegal NCRD or NCBRD Access: Config or IO access of wrong size (e.g. 8 bytes) I2C illegal size EBus Illegal size CSR Illegal Size. Primary Error Status Bit; 1 = Error Received
EBUS_TO_P	26	por_1	0x0	RW1S	EBus Ready Timeout Error Detected. Primary Error Status Bit; 1 = Error Received
MB_PEA_P	25	por_1	0x0	RW1S	Merge Buffer Address Parity Error Detected. FATAL error. Primary Error Status Bit; 1 = Error Received
MB_PER_P	24	por_1	0x0	RW1S	Merge Buffer Parity Error Detected on read. Primary Error Status Bit; 1 = Error Received
MB_PEW_P	23	por_1	0x0	RW1S	Merge Buffer Parity Error Detected on write. Primary Error Status Bit; 1 = Error Received
UE_ASYNC_P	22	por_1	0x0	RW1S	UE Asynchronous Fault. Primary Error Status Bit; 1 = Error Received
CE_ASYNC_P	21	por_1	0x0	RW1S	CE Asynchronous Fault. Primary Error Status Bit; 1 = Error Received
JTE_P	20	por_1	0x0	RW1S	JBus Time-out Err. Fire received a JBus data packet for a noncache- able DMA read with Read Error indicating a Time-out. Primary Error Status Bit; 1 = Error Received

Table 1-74 JBC Error Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
JBE_P	19	por_1	0x0	RW1S	JBus Bus Error. Fire received a JBus data packet for a noncacheable DMA read with Read Error indicating a Bus Error. Primary Error Status Bit; 1 = Error Received
JUE_P	18	por_1	0x0	RW1S	JBus Unmapped Error. This bit gets set when any one of the following conditions is true: <ul style="list-style-type: none"> - Fire received a JBus data packet for a DMA read with Read Error indicating an Unmapped Error. - Fire received a write Packet with an address beyond its range. - Fire received a non 8 byte aligned NCWR for a TSB flush - Fire received NCBWR to 8 MB address space Primary Error Status Bit; 1 = Error Received
IJP_P	17	por_1	0x0	RW1S	Invalid JBUS Port Error. This bit gets set when Fire issues a Write to an invalid JBus port. Primary Error Status Bit; 1 = Error Received
ICISE_P	16	por_1	0x0	RW1S	JBus Illegal Coherency Install State Error. Illegal data install state of 'O'. Primary Error Status Bit; 1 = Error Received
CPE_P	15	por_1	0x0	RW1S	Control Parity Error. FATAL error. Control parity error detected by Fire on the JBus J_PAR line. Only valid if Fire Control and Status Register bit 43 = 1. Primary Error Status Bit; 1 = Error Received
APE_P	14	por_1	0x0	RW1S	Address Parity Error. FATAL error. Address parity error detected by Fire on J_AD bus. Primary Error Status Bit; 1 = Error Received

Table 1-74 JBC Error Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
WR_DPE_P	13	por_1	0x0	RW1S	Write Data Parity Error. Write Data Cycle Parity error detected by Fire on J_AD bus. Primary Error Status Bit; 1 = Error Received
RD_DPE_P	12	por_1	0x0	RW1S	Read Data Parity Error. Read Data Cycle Parity error detected by Fire on JBus J_AD bits. Primary Error Status Bit; 1 = Error Received
ILL_BMW_P	11	por_1	0x0	RW1S	Illegal byte mask detected for NCWR. Primary Error Status Bit; 1 = Error Received
ILL_BMR_P	10	por_1	0x0	RW1S	Illegal byte mask detected for NCRD. Primary Error Status Bit; 1 = Error Received
BJC_P	9	por_1	0x0	RW1S	Bad JBus cmd. Unrecognized JBus command received. Primary Error Status Bit; 1 = Error Received
PIO_UNMAP_P	8	por_1	0x0	RW1S	NCWR or NCBWR Unmapped, A NCWR or NCBWR transaction did not map to any valid address space enabled by Fire. Primary Error Status Bit; 1 = Error Received
PIO_DPE_P	7	por_1	0x0	RW1S	PIO Data Parity Error, Fire detected a Parity Error reading out of the PIO Data Ram in the dmcint block on a NCWR or NCBWR command. Primary Error Status Bit; 1 = Error Received
PIO_CPE_P	6	por_1	0x0	RW1S	PIO Command Parity Error, Fire detected a Parity Error reading out of the PIO Data Ram in the dmcint block on a NCWR, NCBWR, NCRD, or NCBRD command. Primary Error Status Bit; 1 = Error Received

Table 1-74 JBC Error Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
ILL_ACC_P	5	por_1	0x0	RW1S	Illegal NCWR or NCBWR Access: Config or IO access of wrong size (e.g. 8 bytes) I2C illegal size EBus Illegal size CSR Illegal Size. Primary Error Status Bit; 1 = Error Received
UNSOL_RD_P	4	por_1	0x0	RW1S	Unsolicited read response received. Primary Error Status Bit; 1 = Error Received
UNSOL_INTR_P	3	por_1	0x0	RW1S	Unsolicited interrupt ACK/NAK received. Primary Error Status Bit; 1 = Error Received
JTCEEW_P	2	por_1	0x0	RW1S	JBus Time-out Counter Expired on write error. FATAL error. Fire detected a Time-out for a write because the JBus watchdog timer time-out interval was exceeded. Primary Error Status Bit; 1 = Error Received
JTCEEI_P	1	por_1	0x0	RW1S	JBus Time-out Counter Expired on interrupt error. FATAL error. Fire detected a Time-out for an interrupt because the JBus watchdog timer time-out interval was exceeded. Primary Error Status Bit; 1 = Error Received
JTCEER_P	0	por_1	0x0	RW1S	JBus Time-out Counter Expired on read error. FATAL error. Fire detected a Time-out for a read because the JBus watchdog timer time-out interval was exceeded. Primary Error Status Bit; 1 = Error Received

1.3.3.56 JBC Fatal Reset Enable Register (0x00471028/0x0)

This is the JBC Fatal Reset Enable Register. This register is used to enable Fire to generate a Fatal reset on the JBUS when it detects one of the errors in the Fatal error group. This register by default does not enable any of the errors to generate a fatal error so SW will need to enable each error. Fire uses the combination of the bits from the JBC Error Status Clear Register and these enable bits to determine when to cause a fatal reset.

This register is reset by rst_l and not por_l so that Fire will not continuously report Fatal errors after a soft reset. This could happen in the case where the driving flop causing the fatal error is also only cleared by por_l.

Table 1-75 JBC Fatal Reset Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:28				Reserved field
SPARE_P_INT_EN	27:26	rst_l	0x0	RW	Spare, Fatal Reset Enable Bit
MB_PEA_P_INT_EN	25	rst_l	0x0	RW	Merge Buffer Address Parity Error, Fatal Reset Enable Bit
RESERVED	24:16				Reserved field
CPE_P_INT_EN	15	rst_l	0x0	RW	Control Parity Error, Fatal Reset Enable Bit
APE_P_INT_EN	14	rst_l	0x0	RW	Address Parity Error, Fatal Reset Enable Bit
RESERVED	13:7				Reserved field
PIO_CPE_INT_EN	6	rst_l	0x0	RW	PIO Command Parity Error, Fatal Reset Enable Bit
RESERVED	5:3				Reserved field
JTCEEW_P_INT_EN	2	rst_l	0x0	RW	JBus Time-out on write error, Fatal Reset Enable Bit
JTCEEI_P_INT_EN	1	rst_l	0x0	RW	JBus Time-out on interrupt error, Fatal Reset Enable Bit
JTCEER_P_INT_EN	0	rst_l	0x0	RW	JBus Time-out on read error, Fatal Reset Enable Bit

1.3.3.57 JBCINT In Transaction Error Log Register (0x00471030/0x0)

This is the JBCINT In Transaction Error Log Register. This Register is used to capture information when the first primary error is detected by Fire in the JBCINT In group. Please note that until an error in the JBCINT In Group is logged, this register does not contain valid data and will update on every cycle.

Table 1-76 JBCINT In Transaction Error Log Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:56				Reserved field
Q_WORD	55:54	por_1	2'bx	RW	The quad_word offset of the transaction associated with the Error. (i.e. which of the 4 data beats contains the error)
TRANSID	53:48	por_1	6'bx	RW	The transid of the transaction associated with the Error
RESERVED	47:43				Reserved field
ADDRESS	42:0	por_1	43'bx	RW	The address of the transaction associated with the Error

1.3.3.58 JBCINT In Transaction Error Log Register 2 (0x00471038/0x0)

This is the JBCINT In Transaction Error Log Register 2. This Register is used to capture information when the first primary error is detected by Fire in the JBCINT In group. Please note that until an error in the JBCINT In Group is logged, this register does not contain valid data and will update on every cycle.

Table 1-77 JBCINT In Transaction Error Log Register 2

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:52				Reserved field
ARB_WIN	51:28	por_1	24'bx	RW	Last 8 cycles worth of JBus arb winners before the error occurred. 51:49 is the newest cycle. 30:28 is the oldest cycle.

Table 1-77 JBCINT In Transaction Error Log Register 2

Field	Bits	Reset Name	Reset Value	Type	Description
J_REQ	27:21	por_1	7'bx	RW	j_req lines captured when error occurred. It is up to software to determine the mapping between the bits set in this register and the actual corresponding JBus device. device 6 -> device hooked up to i2j_j_req_in_1[5] device 5 -> i2j_j_req_in_1[4] device 4 -> i2j_j_req_in_1[3] device 3 -> i2j_j_req_in_1[2] device 2 -> i2j_j_req_in_1[1] device 1 -> i2j_j_req_in_1[0] device 0 -> Fire
J_PACK	20:0	por_1	21'bx	RW	j_pack lines captured when error occurred. device0 => j_pack[2:0] and device6 => j_pack[20:18]

1.3.3.59 JBCINT Out Transaction Error Log Register (0x00471040 / 0x0)

This is the JBCINT Out Transaction Error Log Register. This Register is used to capture information when the first primary error is detected by Fire in the JBCINT Out group. Please note that until an error in the JBCINT Out Group is logged, this register does not contain valid data and will update on every cycle.

Table 1-78 JBCINT Out Transaction Error Log Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:54				Reserved field
TRANSID	53:48	por_1	6'bx	RW	The transid of the transaction associated with the Error
RESERVED	47:43				Reserved field
ADDRESS	42:0	por_1	43'bx	RW	The address of the transaction associated with the Error

1.3.3.60 JBCINT Out Transaction Error Log Register 2 (0x00471048/0x0)

This is the JBCINT Out Transaction Error Log Register. This Register is used to capture information when the first primary error is detected by Fire in the JBCINT Out group. Please note that until an error in the JBCINT Out Group is logged, this register does not contain valid data and will update on every cycle.

Table 1-79 JBCINT Out Transaction Error Log Register 2

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:52				Reserved field
ARB_WIN	51:28	por_1	24'bx	RW	Last 8 cycles worth of JBus arb winners before the error occurred. 51:49 is the newest cycle. 30:28 is the oldest cycle.
J_REQ	27:21	por_1	7'bx	RW	j_req lines captured when error occurred. It is up to software to determine the mapping between the bits set in this register and the actual corresponding JBus device. device 6 -> device hooked up to i2j_j_req_in_1[5] device 5 -> i2j_j_req_in_1[4] device 4 -> i2j_j_req_in_1[3] device 3 -> i2j_j_req_in_1[2] device 2 -> i2j_j_req_in_1[1] device 1 -> i2j_j_req_in_1[0] device 0 -> Fire
J_PACK	20:0	por_1	21'bx	RW	j_pack lines captured when error occurred. device0 => j_pack[2:0] and device6 => j_pack[20:18]

1.3.3.61 Fatal Error Log Register 1 (0x00471050/0x0)

This is the Fatal Error Log Register 1. This Register is used to capture information when the first primary error is detected by Fire in the Fatal group. Please note that until an error in the Fatal Group is logged, this register does not contain valid data and will update on every cycle.

Table 1-80 Fatal Error Log Register 1

Field	Bits	Reset Name	Reset Value	Type	Description
DATA	63:0	por_1	64'bx	RW	<p>The data in this register is dependent on the type of fatal error which was recorded.</p> <p>For a Merge Buffer PEA Error please refer to the Merge Transaction Error Log Register for field details.</p> <p>For a JBUS APE please refer to the JBCINT Transaction Error Log Register for field details.</p> <p>For a DMCINT Read or Interrupt Timeout Error please refer to the DMCINT IDC Error Log Register for field details.</p> <p>For a DMCINT Write Timeout Error or a JBUS CPE the information stored is not relevant and should be ignored.</p> <p>For a PIO CPE Error please refer to the DMCINT ODCD Error Log Register for field details.</p>

1.3.3.62 Fatal Error Log Register 2 (0x00471058/0x0)

This is the Fatal Error Log Register 2. This Register is used to capture information when the first primary error is detected by Fire in the Fatal group. This will only be used for JBUS CPE and APE Fatal Errors. For all other Fatal errors this register will not contain valid data and should be ignored.

Please note that until a JBUS CPE or APE error is logged, this register does not contain valid data and will update on every cycle.

Table 1-81 Fatal Error Log Register 2

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:52				Reserved field
ARB_WIN	51:28	por_1	24'bx	RW	Last 8 cycles worth of JBus arb winners before the error occurred. 51:49 is the newest cycle. 30:28 is the oldest cycle.
J_REQ	27:21	por_1	7'bx	RW	j_req lines captured when error occurred. It is up to software to determine the mapping between the bits set in this register and the actual corresponding JBus device.
J_PACK	20:0	por_1	21'bx	RW	j_pack lines captured when error occurred. device0 => j_pack[2:0] and device6 => j_pack[20:18]

1.3.3.63 Merge Transaction Error Log Register (0x00471060/0x0)

This is the Merge Transaction Error Log Register. This Register is used to capture information when the first primary error is detected by Fire in the Merge group. Please note that until an error in the Merge Group is logged, this register does not contain valid data and will update on every cycle.

Table 1-82 Merge Transaction Error Log Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:56				Reserved field
Q_WORD	55:54	por_1	2'bx	RW	The quad_word offset of the transaction associated with the Error. (i.e. which of the 4 data beats contains the error)
TRANSID	53:48	por_1	6'bx	RW	The transid of the transaction associated with the Error
JBC_TAG	47:43	por_1	5'bx	RW	The JBC tag of the transaction associated with the Error

Table 1-82 Merge Transaction Error Log Register

Field	Bits	Reset Name	Reset Value	Type	Description
ADDRESS	42:0	por_1	43'bx	RW	The address of the transaction associated with the Error

1.3.3.64 DMCINT ODCD Error Log Register (0x00471068 / 0x0)

This is the DMCINT ODCD Transaction Error Log Register. This Register is used to capture information when the first primary error is detected by Fire in the DMCINT ODCD group. Please note that until an error in the DMCINT ODCD Group is logged, this register does not contain valid data and will update on every cycle.

Table 1-83 DMCINT ODCD Error Log Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:54				Reserved field
TRANS_ID	53:52	por_1	2'bx	RW	Read Transaction ID associated with the Error, if error was due to a Read Request
AID	51:48	por_1	4'bx	RW	Agent ID associated with the Error
TRANS_TYPE	47:43	por_1	5'bx	RW	JBUS Transaction Type associated with the Error
ADDRESS	42:0	por_1	43'bx	RW	The address of the transaction associated with the Error

1.3.3.65 DMCINT IDC Error Log Register (0x00471070 / 0x0)

This is the DMCINT IDC Transaction Error Log Register. This Register is used to capture information when the first primary error is detected by Fire in the DMCINT IDC group. Please note that until an error in the DMCINT IDC Group is logged, this register does not contain valid data and will update on every cycle.

The DMC_CTAG is as follows for the below transactions

RDD transactions

[15] type 1'b0 - indicates DMA/INT transaction

[10:6] dptr[4:0] DMC-DOU DATA-BUFFER destination address

[5:1] pkt_tag[4:0] DMC-PSB address for associated READ packet

[0] cl_sts 1'b1 - 1st cacheline request for a DMC-DMA_RD packet

RDS transactions

[15] type 1'b1 - indicates MMU tablewalk transaction
 [10:6] rsv[4:0] reserved
 [5:0] mtag[5:0] MMU tablewalk tracking ID
 INT transactions
 [15] type 1'b0 - indicates DMA/INT transaction
 [10:3] rsv[7:0] reserved
 [2:1] mdo_tag[1:0] IMU Group_Controller ID
 [0] rsv reserved

Table 1-84 DMCINT IDC Error Log Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:28				Reserved field
DMC_CTAG	27:16	por_1	12'bx	RW	{DMC_CTAG[15], DMC_CTAG[10:0]} for READ/INT timeout
TRANSID	15:14	por_1	2'bx	RW	READ_TRANSID if error was unsolicited READ_RETURN or READ
AGNTID	13:10	por_1	4'bx	RW	AGNTID of READ if error was unsolicited READ_RETURN or READ
SRCID	9:5	por_1	5'bx	RW	SRCID of INT if error was unsolicited INT_(N)ACK or INT timeout
TARGID	4:0	por_1	5'bx	RW	TARGID of INT if error was unsolicited INT_(N)ACK or INT

1.3.3.66 CSR Error Log Register (0x00471078 / 0x0)

This is the CSR Error Log Register. This Register is used to capture information when the first primary error is detected by Fire in the CSR group. Please note that until an error in the CSR Group is logged, this register does not contain valid data and will update on every cycle.

Table 1-85 CSR Error Log Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:43				Reserved field
WRITE	42	por_1	1'bx	RW	If the transaction was a read or a write. 1 = write 0 = read
BMASK	41:26	por_1	16'bx	RW	The Byte mask of the transaction associated with the Error
ADDRESS	25:0	por_1	26'bx	RW	The address of the transaction associated with the Error

1.3.3.67 JBC Core and Block Interrupt Enable Register (0x00471800 / 0x0)

This is the JBC Core and Block Interrupt Enable Register. This register is used enable interrupts at a block and core level. This register allows SW to mask interrupts it does not want to see with a core and block level granularity. Each of the supported JBC errors falls under one of the block enables. Also note that these are a smaller subset of groups than found in the JBC Error Log Enable Section. The Fatal error group as well as the sub set break downs for the DMCINT and the JBUSINT have been removed and the error put into their originating large blocks

CSR : ebus_to

MERGE: mb_pea, mb_per, mb_pew

DMCINT: unsol_rd, unsol_intr, pio_unmap, pio_dpe, ill_acc, jtceei, jtceer, jtceew, pio_cpe, spare bit 2

JBUSINT: ue_asyn, ce_asyn, jte, jbe, jue, ijp, icise, cpe, ape, wr_dpe, rd_dpe, ill_bmw, ill_bmr, bjc, spare bit 0 and 1

If a particular block is not enabled all errors from that block will not generate an interrupt no matter what the individual error interrupt enable is set to. If the JBC core interrupt enable is not enabled all JBC errors will not generate an interrupt.

Please note, that if a condition occurs that should have caused an interrupt via mondo 63 and a particular enable is disabled at that time (not causing the mondo to be sent), if that enable is then enabled before clearing the stored offending condition an interrupt via mondo 63 will then be sent.

Table 1-86 JBC Core and Block Interrupt Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
JBC	63	rst_1	0x0	RW	The enable bit for the JBC interrupt. 0 = Not Enabled, 1 = Enabled
RESERVED	62:4				Reserved field
CSR	3	rst_1	0x0	RW	The enable bit for the csr interrupt. 0 = Not Enabled, 1 = Enabled
MERGE	2	rst_1	0x0	RW	The enable bit for the merge interrupt. 0 = Not Enabled, 1 = Enabled
JBCINT	1	rst_1	0x0	RW	The enable bit for the jbc_int interrupt. 0 = Not Enabled, 1 = Enabled
DMCINT	0	rst_1	0x0	RW	The enable bit for the dmc_int interrupt. 0 = Not Enabled, 1 = Enabled

1.3.3.68 JBC Core and Block Error Status Register (0x00471808 / 0x0)

This is the JBC Core and Block Error Status Register. This is a read only register which SW uses to determine which block in the JBC core caused an interrupt when it received a mondo 63. A value of 1 means that particular block caused an interrupt to be generated

Table 1-87 JBC Core and Block Error Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:4				Reserved field
CSR	3	rst_1	0x0	R	The CSR block has an interrupt that needs to be processed
MERGE	2	rst_1	0x0	R	The merge has an interrupt that needs to be processed
JBCINT	1	rst_1	0x0	R	The jbcint has an interrupt that needs to be processed

Table 1-87 JBC Core and Block Error Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
DMCINT	0	rst_l	0x0	R	The dmcint has an interrupt that needs top be processed

1.3.3.69 JBC Performance Counter Select Register (0x00472000 / 0x0)

This is the JBC Performance Counter Select Register. This register provides the select inputs for both of the performance counters in the JBC core. These selects are used to select which of the 24 possible events each counter should count. The counters both default to disabled. When selecting an event to count, the counting will begin as soon as this register is updated. It should be noted that in changing the value of the select, the counter register DOES NOT reset to 0. It will continue to count at its current value. If the counter needs to start at 0, it should be cleared before the new value of the select is written.

Table 1-88 JBC Performance Counter Select Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:16				Reserved field
SEL1	15:8	rst_l	0x0	RW	The values for Select 1 are the same as select 0

Table 1-88 JBC Performance Counter Select Register

Field	Bits	Reset Name	Reset Value	Type	Description
SEL0	7:0	rst_l	0x0	RW	Select for counter 0: 00 = None 01 = Clock cycles 02 = JBUS Idle Time in cycles 03 = Time Fire is on the JBus in cycles 04 = Total Read latency for sampled Read transactions in cycles 05 = Number of Reads Sampled for above Read Latency 06 = Number of PIO's to the I2C 07 = Number of PIO's to the EBus 08 = Number of PIO's to the CSR Ring A 09 = Number of PIO's to the CSR Ring B 10 = Number of Partial Writes 11 = Number of Total Writes 12 = Number of Total Reads 13 = Number of Cycles Fire Drives AOKOFF 14 = Number of Cycles Fire Drives DOKOFF 15 = Number of Cycles Fire Drives DOKOFF or AOKOFF 16 = Number of JBUS Coherent Transactions 17 = Number of Fire's Coherent Transactions 18 = Number of JBUS Non Coherent Transactions 19 = Number of Foreign PIO Hits 20 = Number of WB Issued by Fire 21 = Number of PIO Writes to PCIE A 22 = Number of PIO Reads to PCIE A 23 = Number of PIO Writes to PCIE B 24 = Number of PIO Reads to PCIE B

1.3.3.70 JBC Performance Counter Zero Register (0x00472008 / 0x0)

This is the JBC Performance Counter Zero Register. This register is used to read and write the value of Counter Zero for the JBC core. This counter will increment once for every clock cycle that the selected event chosen by the JBC Performance Counter Select Register occurs. This value can be set and cleared by SW.

Table 1-89 JBC Performance Counter Zero Register

Field	Bits	Reset Name	Reset Value	Type	Description
CNT	63:0	rst_l	0x0	RW	Counter

1.3.3.71 JBC Performance Counter One Register (0x00472010 / 0x0)

This is the JBC Performance Counter One Register. This register is used to read and write the value of Counter one for the JBC core. This counter will increment once for every clock cycle that the selected event chosen by the JBC Performance Counter Select Register occurs. This value can be set and cleared by SW.

Table 1-90 JBC Performance Counter One Register

Field	Bits	Reset Name	Reset Value	Type	Description
CNT	63:0	rst_l	0x0	RW	Counter

1.3.3.72 Fire and JBC Debug Select Register A (0x00473000 / 0x0)

This is Fire and JBC Debug Select Register A. This register is used to select which of the many debug signals will be sent to Fire's Debug port A. This debug port will update on every JBUS clock. Debug ports are only intended for HW lab debug use.

Table 1-91 Fire and JBC Debug Select Register A

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:12				Reserved field
CORE_SEL	11:10	rst_l	0x0	RW	Fire 200 MHz Core Debug Selects for Port A 00 - All Zeros 01 - JBC 10 - DMC A 11 - DMC B
RESERVED	9				Reserved field

Table 1-91 Fire and JBC Debug Select Register A

Field	Bits	Reset Name	Reset Value	Type	Description
BLOCK_SEL	8:6	rst_1	0x0	RW	JBC Block Debug Selects for Port A 000 - All Zeros 001 - Logic Analyzer Verification Algorithm 010 - DMCINT Block Selects 011 - JBCINT Block Selects 100 - Merge Block Selects 101 - Reset Block Selects 110 - Snoop Block Selects 111 - CSR Block Selects
SUB_SEL	5:3	rst_1	0x0	RW	Select JBC Sub-Block for Port A
SIGNAL_SEL	2:0	rst_1	0x0	RW	Select the signals for Port A

1.3.3.73 Fire and JBC Debug Select Register B (0x00473008 / 0x0)

This is Fire and JBC Debug Select Register B. This register is used to select which of the many debug signals will be sent to Fires Debug port B. This debug port will update on every JBUS clock. Debug ports are only intended for HW lab debug use.

Table 1-92 Fire and JBC Debug Select Register B

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:12				Reserved field
CORE_SEL	11:10	rst_1	0x0	RW	Fire 200 MHz Core Debug Selects for Port B 00 - All Zeros 01 - JBC 10 - DMC A 11 - DMC B
RESERVED	9				Reserved field

Table 1-92 Fire and JBC Debug Select Register B

Field	Bits	Reset Name	Reset Value	Type	Description
BLOCK_SEL	8:6	rst_l	0x0	RW	JBC Block Debug Selects for Port B 000 - All Zeros 001 - Logic Analyzer Verification Algorithm 010 - DMCINT Block Selects 011 - JBCINT Block Selects 100 - Merge Block Selects 101 - Reset Block Selects 110 - Snoop Block Selects 111 - CSR Block Selects
SUB_SEL	5:3	rst_l	0x0	RW	Select JBC Sub-Block for Port B
SIGNAL_SEL	2:0	rst_l	0x0	RW	Select the signals for Port B

1.3.3.74 I2C Bus-x Slave Address Register (0x00520000, 0x00530000 / 0x0)

For 7-bit addressing, SLA6 – SLA0 is the 7-bit address of the I2C when in slave mode. When the I2C receives this address after a START condition, it will generate an interrupt and enter slave mode. (SLA6 corresponds to the first bit received from the I2C bus.) If GCE is set to one, the I2C will also recognize the general call address (0x0).

Table 1-93 I2C x Slave Address Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:8				Reserved field
SLA[6:0] or 5'b11110,SLAX9,SLAX8	7:1	rst_l	0x0	RW	Slave Address
GCE	0	rst_l	0x0	RW	General Call Address Enable

1.3.3.75 I2C Bus-x Extended Slave Address Register (0x00520008, 0x00530008 / 0x0)

For 10-bit addressing, when the address received, SLA[6:2], starts with 0x1E, the I2C recognizes this as the first part of a 10-bit address. If the next two bits match I2C Slave Address Register[2:1] (i.e. SLAX9 and SLAX8 of the device's extended address), it sends an ACK. (The device does not generate an interrupt at this point.) After the next byte of the address has been received and if the address matches, the I2C generates an interrupt and goes into slave mode.

Table 1-94 I2C x Extended Slave Address Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:8				Reserved field
SLAX[7:0]	7:0	rst_l	0x0	RW	Extended Slave Address

1.3.3.76 I2C Bus-x Data Byte Register (0x00520010, 0x00530010 / 0x0)

Table 1-95 I2C x Data Byte Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:8				Reserved field
DATA	7:0	rst_l	0x0	RW	DATA contains the data byte/slave address to be transmitted or the data byte that has just been received. In transmit mode, the byte is sent MSB first; in receive mode, the first bit received will be placed in the MSB of the DATA. After each byte is transmitted, DATA will contain the byte that was actually present on the bus so in the event of lost arbitration, it will contain the received byte.

1.3.3.77 I2C Bus-x Control Register (0x00520018, 0x00530018 / 0x0)

Table 1-96 I2C x Control Register (Sheet 1 of 3)

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:8				Reserved field
IEN	7	rst_l	0x0	RW	1'b0 => Interrupt disabled (the interrupt line will always remain low) 1'b1 => Interrupt Enabled (the interrupt line will go high when the IFLG bit is set).

Table 1-96 I2C x Control Register (Sheet 2 of 3)

Field	Bits	Reset Name	Reset Value	Type	Description
ENAB	6	rst_l	0x0	RW	<p>1'b0 => the I2C bus inputs ISDA/ISCL are ignored and the I2C will not respond to any address on the bus.</p> <p>1'b1 => the I2C will respond to calls to its slave address – and to the general call address if the GCE bit in the ADDR register is set.</p>
STA	5	rst_l	0x0	RW	<p>When STA is set to one, the I2C enters master mode and will transmit a START condition on the bus when the bus is free. If the STA bit is set to one when the I2C is already in master mode and one or more bytes have been transmitted, then a repeated START condition will be sent. If the STA bit is set to one when the I2C is being accessed in slave mode, the I2C will complete the data transfer in slave mode then enter master mode when the bus has been released. The STA bit is cleared automatically after a START condition has been sent: writing a zero to this bit has no effect.</p>
STP	4	rst_l	0x0	RW	<p>If STP is set to one in master mode, a STOP condition is transmitted on the I2C bus. If the STP bit is set to one in slave mode, the I2C will behave as if a STOP condition has been received, but no STOP condition will be transmitted on the I2C bus. If both STA and STP bits are set, the I2C will first transmit the STOP condition (if in master mode) then transmit the START condition. The STP bit is cleared automatically: writing a zero to this bit has no effect.</p>
IFLG	3	rst_l	0x0	RW	<p>IFLG is automatically set to one when any of 28 (out of the possible 29) I2C states is entered. The only state that does not set IFLG is state 0xF8 (see STAT Register section). If IFLG is set to one and the IEN bit is set, the interrupt line goes high. When IFLG is set by the I2C, the low period of the I2C bus clock line (SCL) is stretched and the data transfer is suspended. When zero is written to IFLG, the interrupt line goes low and the I2C clock line is released.</p>

Table 1-96 I2C x Control Register (Sheet 3 of 3)

Field	Bits	Reset Name	Reset Value	Type	Description
AAK	2	rst_1	0x0	RW	<p>When AAK is set to one, an Acknowledge (low level on SDA) will be sent during the acknowledge clock pulse on the I2C bus if:</p> <ul style="list-style-type: none"> i Either the whole of a matching 7-bit slave address or the first or the second byte of a matching 10-bit slave address has been received. ii The general call address has been received and the GCE bit in the ADDR register is set to one. iii A data byte has been received in master or slave mode. <p>When AAK is cleared to zero, a Not Acknowledge (high level on SDA) will be sent when a data byte is received in master or slave mode. If AAK is cleared to zero in the slave transmitter mode, the byte in the DATA register is assumed to be the 'last byte'. After this byte has been transmitted, the I2C will enter state 0xC8 then return to the idle state. The I2C will not respond as a slave unless AAK is set.</p>
RESERVED	1:0				Reserved field

1.3.3.78 I2C Bus-x Status Register (0x00520020, 0x00530020 / 0xF8)

Table 1-97 I2C x Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:8				Reserved field

Table 1-97 I2C x Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
STATUS	7:0	rst_l	0xF8	R	0x00 => Bus error 0x08 => START condition transmitted 0x10 => Repeated START condition transmitted 0x18 => Address + Write bit transmitted, ACK received 0x20 => Address + Write bit transmitted, ACK not received 0x28 => Data byte transmitted in master mode, ACK received 0x30 => Data byte transmitted in master mode, ACK not received 0x38 => Arbitration lost in address or data byte 0x40 => Address + Read bit transmitted, ACK received 0x48 => Address + Read bit transmitted, ACK not received 0x50 => Data byte received in master mode, ACK transmitted 0x58 => Data byte received in master mode, not ACK transmitted 0x60 => Slave address + Write bit received, ACK transmitted 0x68 => Arbitration lost in address as master, slave address + Write bit received, ACK transmitted 0x70 => General Call address received, ACK transmitted 0x78 => Arbitration lost in address as master, General Call address received, ACK transmitted 0x80 => Data byte received after slave address received, ACK transmitted 0x88 => Data byte received after slave address received, not ACK transmitted 0x90 => Data byte received after General Call received, ACK transmitted 0x98 => Data byte received after General Call received, not ACK transmitted 0xA0 => STOP or repeated START condition received in slave mode 0xA8 => Slave address + Read bit received, ACK transmitted 0xB0 => Arbitration lost in address as master, slave address + Read bit received, ACK transmitted 0xB8 => Data byte transmitted in slave mode, ACK received 0xC0 => Data byte transmitted in slave mode, ACK not received 0xC8 => Last byte transmitted in slave mode, ACK received 0xD0 => Second Address byte + Write bit transmitted, ACK received 0xD8 => Second Address byte + Write bit transmitted, ACK not received 0xE0 => Unused 0xE8 => Unused 0xF0 => Unused 0xF8 => No relevant status information, IFLG=0

1.3.3.79 I2C Bus-x Clock Control Register (0x00520028, 0x00530028 / 0x0)

This register controls the frequency at which the I2C bus is sampled and the frequency of the I2C clock line (SCL) when the I2C is in master mode. The input clock frequency (of CLK) is first divided by a factor of 2^N , where N is the value defined by bits 2 – 0 of CCR. The output of this clock divider is F_0 . F_0 is then divided by a further factor of M+1, where M is the value defined by bits 6 – 3 of CCR. The output of this clock divider is F_1 .

The I2C bus is sampled by the I2C at the frequency defined by F_0 :

$$F_{SAMP} = F_0 = F_{CLK} / 2^N$$

The I2C OSCL output frequency, in master mode, is $F_1 / 10$:

$$F_{OSCL} = F_1 / 10 = F_{CLK} / (2^N \cdot (M + 1) \cdot 10)$$

The use of two separately programmable dividers allows the master mode output frequency to be set independently of the frequency at which the I2C bus is sampled. This is particularly useful in multi-master systems because the frequency at which the I2C bus is sampled must be at least 10 times the frequency of the fastest master on the bus to ensure that START and STOP conditions are always detected. By using two programmable clock divider stages, a high sampling frequency can be ensured while allowing the master mode output to be set to a lower frequency.

Table 1-98 I2C x Clock Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:7				Reserved field
M	6:3	rst_1	0x0	W	See description above.
N	2:0	rst_1	0x0	W	See description above.

1.3.3.80 I2C Bus-x Software Reset Register (0x00520030, 0x00530030 / 0x0)

On a hardware reset, The Address, Extended Address, Data, and Control Registers are cleared to 0x0, the Status Register is set to 0xF8, and the Clock Control Register is set to 00h. A software reset sets the I2C back to idle, and sets the STP, STA and IFLG bits in the Control register to zero.

Table 1-99 I2C x Software Reset Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:8				Reserved field
RESET	7:0	rst_1	0x0	RW	Any writes to this register will software reset this I2C controller.

1.3.4 PCI-E Registers

1.3.4.1 Interrupt Mapping Registers (0x00601000-0x006011F8, 0x00701000-0x007011F8 / 0x0)

The Interrupt Mapping Registers comprise a series of 64 consecutive registers, one for each Mondo Interrupt. They are read/write registers software uses to set up each of the 64 supported mondos. Through these registers, software, on a per-mondo basis, can set up the Mondo Mode and JPID of the target CPU the mondo is destined for, as well as enable or disable the mondo's valid bit. Software can also select which Interrupt Controller (group controller) the mondo will use.

Table 1-100 Interrupt Mapping Registers

Field	Bits	Reset Name	Reset Value	Type	Description
MDO_MODE	63	rst_l	0x0	RW	This bit is used to select which of two formats the mondo will use: 1 = data-bearing mondo 0 = non-data-bearing mondo (normal mondo). The value of this bit, will be used as bit 63 of the first data word in the mondo vector. In general, EQ mondos should have this bit set to 1 and non-EQ mondos should set this bit to 0.
RESERVED	62:32				Reserved field
V	31	rst_l	0x0	RW	Valid bit. When set to 0, an interrupt will not be dispatched to a CPU from this PCIE leaf. Has no other impact on interrupt state.
T_JPID	30:26	rst_l	0x0	RW	JPID of the processor that this interrupt will be sent to.
RESERVED	25:10				Reserved field

Table 1-100 Interrupt Mapping Registers

Field	Bits	Reset Name	Reset Value	Type	Description
INT_CNTRL_NUM	9:6	rst_l	0x0	RW	Interrupt Controller Number. This is used to select which interrupt controller will issue the interrupt. Only 1 bit may be selected at a time. Valid Values are as follows: 0000 - No controller selected 0001 - Interrupt Controller 0 0010 - Interrupt Controller 1 0100 - Interrupt Controller 2 1000 - Interrupt Controller 3 Other values are a programming error, and the results are undefined
RESERVED	5:0				Reserved field

1.3.4.2 Interrupt Clear Registers (0x00601400-0x006015F8, 0x00701400-0x007015F8 / 0x0)

The Interrupt Clear Registers comprise a series of 64 consecutive registers, one for each Mondo Interrupt. They are read/write registers which software uses to read and write the state of the 64 supported mondos. Through these registers, software, on a per-mondo basis, can at any time obtain the current value of the mondo state or direct the mondo into one of the legal states.

Table 1-101 Interrupt Clear Registers

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:2				Reserved field

Table 1-101 Interrupt Clear Registers

Field	Bits	Reset Name	Reset Value	Type	Description
INT_STATE	1:0	rst_l	0x0	RW	<p>Interrupt State. When writing to this register the lower two bits control the state bits for the interrupt state machine associated with this interrupt. The following values may be written:</p> <ul style="list-style-type: none"> 00 - Set the state machine to the IDLE state. 01 - Set the state machine to the RECEIVED state 10 - Reserved. Use of this value is a programming error, and the results are undefined. 11 - Set the state machine to the PENDING state. <p>When reading from this register, the actual state of the associated interrupt state machine is read. The legal values are the same as listed above.</p>

1.3.4.3 Interrupt Retry Timer Register (0x00601A00, 0x00701A00 / 0x0)

The Interrupt Retry Timer Register is a read/write register software uses to set up the value of the Interrupt Retry Timer used by all four of the Interrupt Controllers. When the IMU receives a NACK for a given mondo, the value from this register is loaded into the interrupt controller counter which then begins to decrement the value by one on every clock cycle. When the value of the counter reaches zero, the mondo which was NACK'D is then retried. A value of all 0's means that the mondo will be retried on the very next cycle.

Table 1-102 Interrupt Retry Timer Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:25				Reserved field
LIMIT	24:0	rst_l	0x0	RW	Limit the retry interval in clock cycles (JBUS FREQ)

1.3.4.4 Interrupt State Status Register 1 (0x00601A10, 0x00701A10 / 0x0)

Interrupt State Status Register 1 is a read-only register software uses to read the state of the first 32 mondos (0-31) supported by Fire in a single 64-bit access. Each mondo is represented by 2 bits where the same state encodings from the Interrupt Clear Registers are used.

Table 1-103 Interrupt State Status Register 1

Field	Bits	Reset Name	Reset Value	Type	Description
STATE	63:0	rst_l	0x0	R	State Values for Mondos 0 through 31 Each state is 2 bits in the register with the MSB being the 2nd bit of Mondo 31 and the LSB being the 1st bit of Mondo 0

1.3.4.5 Interrupt State Status Register 2 (0x00601A18, 0x00701A18 / 0x0)

Interrupt State Status Register 2 is a read only register software uses to read the state of the second 32 mondos (32-63) supported by Fire in a single 64-bit access. Each mondo is represented by 2 bits where the same state encodings from the Interrupt Clear Registers are used.

Table 1-104 Interrupt State Status Register 2

Field	Bits	Reset Name	Reset Value	Type	Description
STATE	63:0	rst_l	0x0	R	State Values for Mondos 32 through 63 Each state is 2 bits in the register with the MSB being the 2nd bit of Mondo 63 and the LSB being the 1st bit of Mondo 32

1.3.4.6 INTX Status Register (0x0060B000, 0x0070B000 / 0x0)

The INTX Status Register is a read-only register software uses to obtain the status of the four emulated INTX interrupts in Fire. Each of the four bits will be updated to reflect the current hardware status. (The status is changed by the reception of either Assert or Deassert PCIE messages or a write by software to the any of the four INT X Clear Registers.)

Table 1-105 INTX Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:4				Reserved field

Table 1-105 INTX Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
INT_A	3	rst_1	0x0	R	INT_A Status: 0 = No INT_A, 1 = INT_A This register will be set when an assert INT_A message is received and will be cleared when a deassert INT_A message is received or when cleared via the INT_A Clear Register by software.
INT_B	2	rst_1	0x0	R	INT_B Status: 0 = No INT_B, 1 = INT_B This register will be set when an assert INT_B message is received and will be cleared when a deassert INT_B message is received or when cleared via the INT_B Clear Register by software.
INT_C	1	rst_1	0x0	R	INT_C Status: 0 = No INT_C, 1 = INT_C. This register will be set when an assert INT_C message is received and will be cleared when a deassert INT_C message is received or when cleared via the INT_C Clear Register by software.
INT_D	0	rst_1	0x0	R	INT_D Status: 0 = No INT_D, 1 = INT_D This register will be set when an assert INT_D message is received and will be cleared when a deassert INT_D message is received or when cleared via the INT_D Clear Register by software.

1.3.4.7 INT A Clear Register (0x0060B008, 0x0070B008 / 0x0)

The INT A Clear Register is a Read/Write-One-to-Clear register software uses to complete two tasks. First, software can obtain the current hardware status of INT A just as in bit 3 of the INTX Status Register above. Second, software can clear the INT A interrupt by writing a value of 1'b1 to bit 0 of this register. This functionality is required in case the PCI-E Link goes down; software must clear the INT_A bit in order

to maintain consistent INTx state on the fabric for that port. These registers should only be used for that purpose and NEVER during normal operation. If they are used results are undefined and interrupts MAY BE LOST.

Table 1-106 INT A Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:1				Reserved field
CLR	0	rst_1	0x0	RW1C	Write 0 has no effect. Write 1 will clear the INT_A bit of the INTx Status Register. When reading, the value of the INT_A bit from the INTx Status Register will be returned.

1.3.4.8 INT B Clear Register (0x0060B010, 0x0070B010 / 0x0)

The INT B Clear Register is a Read/Write-One-to-Clear register software uses to complete two tasks. First, software can obtain the current hardware status of INT_B just as in bit 3 of the INTx Status Register above. Second, software can clear the INT_B interrupt by writing a value of 1'b1 to bit 0 of this register. This functionality is required in case the PCI-E Link goes down; software must clear the INT_B bit in order to maintain consistent INTx state on the fabric for that port. These registers should only be used for that purpose and NEVER during normal operation. If they are used results are undefined and interrupts MAY BE LOST.

Table 1-107 INT B Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:1				Reserved field
CLR	0	rst_1	0x0	RW1C	Write 0 has no effect. Write 1 will clear the INT_B bit of the INTx Status Register. When reading, the value of the INT_B bit from the INTx Status Register will be returned.

1.3.4.9 INT C Clear Register (0x0060B018, 0x0070B018 / 0x0)

The INT C Clear Register is a Read/Write-One-to-Clear register software uses to complete two tasks. First, software can obtain the current hardware status of INT_C just as in bit 3 of the INTx Status Register above. Second, software can clear the INT_C interrupt by writing a value of 1'b1 to bit 0 of this register. This functionality is required in case the PCI-E Link goes down; software must clear the INT_C bit in order

to maintain consistent INTx state on the fabric for that port. These registers should only be used for that purpose and NEVER during normal operation. If they are used results are undefined and interrupts MAY BE LOST.

Table 1-108 INT C Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:1				Reserved field
CLR	0	rst_1	0x0	RW1C	Write 0 has no effect. Write 1 will clear the INT_C bit of the INTx Status Register. When reading, the value of the INT_C bit from the INTx Status Register will be returned.

1.3.4.10 INT D Clear Register (0x0060B020, 0x0070B020 / 0x0)

The INT_D Clear Register is a Read/Write-One-to-Clear register which software uses to complete two tasks. First, software can obtain the current hardware status of INT_D just as in bit 3 of the INTx Status Register above. Second, software can clear the INT_D interrupt by writing a value of 1'b1 to bit 0 of this register. This functionality is required in case the PCI-E Link goes down; software must clear the INT_D bit in order to maintain consistent INTx state on the fabric for that port. These registers should only be used for that purpose and NEVER during normal operation. If they are used results are undefined and interrupts MAY BE LOST.

Table 1-109 INT D Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:1				Reserved field
CLR	0	rst_1	0x0	RW1C	Write 0 has no effect. Write 1 will clear the INT_D bit of the INTx Status Register. When reading, the value of the INT_D bit from the INTx Status Register will be returned.

1.3.4.11 Event Queue Base Address Register (0x00610000, 0x00710000 / 0x0)

The Event Queue Base Address Register is used as the base address for all Event Queue writes issued by Fire. It is the start of a 512K aligned contiguous memory region containing thirty six 8K pages, one page per event queue. This register can be set up to

have EQ writes use a physical address or a virtual address which needs to be translated by the MMU. In either case the address must be set up correctly. Please refer to the addressing sections of the PRM for details.

Table 1-110 Event Queue 15 Address Register

Field	Bits	Reset Name	Reset Value	Type	Description
ADDRESS	63:19	rst_l	0x0	RW	EQ Base Address, 512K Aligned This address must be a properly formatted physical or virtual address. For a bypass address, the upper 14 address bits [63:50] must be set to all 1's, address bits [49:43] must be set to zero, and address bit [42] must be set to zero. The lower bits of the address [41:18] are used as the cacheable physical address on JBus For a virtual address, bit 63 needs to be zero, bits [62:32] are don't-care, and bits [31:18] are used to access the MMU.
RESERVED	18:0				Reserved field

1.3.4.12 Event Queue Control Set Register (0x00611000-0x00611118, 0x00711000-0x00711118 / 0x0)

The Event Queue Control Set Registers comprise 36 consecutive registers, one for each EQ. These are write-only registers software uses to complete two tasks. First, software uses these registers to make an EQ active. The EQ will move from the IDLE to ACTIVE state when a 1 is written to the 'EN' bit. Second, software uses this register to inject a EQ overflow Error. When a 1 is written to the 'ENOVERR' bit the overflow error status bit for the EQ will be set and it will transition to the ERROR state.

Table 1-111 Event Queue Control Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:58				Reserved field
ENOVERR	57	rst_l	0x0	L	0 = No action; 1 = Set OVERR bit. A read of this register is not allowed. This bit should only be set if the EQ is currently in the ACTIVE state. Setting this bit when the EQ is IDLE will cause undetermined results.

Table 1-111 Event Queue Control Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	56:45				Reserved field
EN	44	rst_l	0x0	L	0 = No action; 1 = Enable EQ. EQ will be running when STATE = ACTIVE. A read of this register is not allowed. This bit should only be written when the EQ is currently IDLE. If the EQ is not in the IDLE state this operation will have no effect on the state of the EQ.
RESERVED	43:0				Reserved field

1.3.4.13 Event Queue Control Clear Register (0x00611200-0x00611318, 0x00711200-0x00711318 / 0x0)

The Event Queue Control Clear Registers comprise 36 consecutive registers, one associated with each EQ. These are write-only registers software uses to complete three tasks. First, for a given EQ, software uses the associated register to disable the EQ. The EQ will move from the ACTIVE to IDLE state when a one is written to the 'DIS' bit in the associated register. Second, again for a given EQ, software uses the register to clear an EQ overflow error. When a one is written to the 'COVERR' bit in the associated register, the overflow error status bit for that EQ will be cleared. Third, for a given EQ, software uses the register to take an EQ from the ERROR to the IDLE state by writing a one to the 'E2I' bit

Table 1-112 Event Queue Control Clr Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:58				Reserved field
COVERR	57	rst_l	0x0	L	0-no action; 1-Clear OVERR bit, A read of this register is not allowed
RESERVED	56:48				Reserved field

Table 1-112 Event Queue Control Clr Register

Field	Bits	Reset Name	Reset Value	Type	Description
E2I	47	rst_1	0x0	L	0-no action;1-go from ERROR to IDLE. A read of this register is not allowed. This bit should only be written when the EQ is currently in the ERROR state. If the EQ is not in the ERROR state this operation will have no effect on the state of the EQ.
RESERVED	46:45				Reserved field
DIS	44	rst_1	0x0	L	0-no action; 1-Disable EQ. A read of this register is not allowed. This bit should only be set if the EQ is currently in the ACTIVE state. If the EQ is not in the ACTIVE state this operation will have no effect on the state of the EQ.
RESERVED	43:0				Reserved field

1.3.4.14 Event Queue State Register (0x00611400-0x00611518, 0x00711400-0x00711518 / 0x1)

The Event Queue State Registers comprise 36 consecutive registers, one for each EQ. They are read-only registers software uses to determine the current state of a given Event Queue. Each EQ is represented by a 3 bit 1-hot encoding to represent each of the three possible EQ state values: 001-IDLE, 010-ACTIVE, 100-ERROR

Table 1-113 Event Queue State Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:3				Reserved field
STATE	2:0	rst_1	0x1	R	Event Queue State 001-IDLE, 010-ACTIVE, 100-ERROR

1.3.4.15 Event Queue Tail Register (0x00611600-0x00611718, 0x00711600-0x00711718 / 0x0)

The Event Queue Tail Registers comprise 36 consecutive registers, one for each EQ. These registers should be considered READ ONLY registers for software which uses them to complete two tasks. First, software uses these registers to obtain the value of the current hardware tail pointer for a given EQ. Second, software uses these registers to obtain the value of a given EQ overflow Error bit.

Please note that even though the tail pointer field is writable by software it should not be written except for diagnostic purposes. Writing this field in normal operation will cause unpredictable results and loss of interrupts. It is also possible for software to initialize the tail pointer to a value other than 0, but this is unnecessary and only should be done with good reason.

Table 1-114 Event Queue Tail Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:58				Reserved field
OVERR	57	rst_1	0x0	R	1-EQ Overflow occurred.
RESERVED	56:7				Reserved field
TAIL	6:0	rst_1	0x0	RW	Value of the current HW tail pointer. In normal operation it is read by SW and written by HW.

1.3.4.16 Event Queue Head Register (0x00611800-0x00611918, 0x00711800-0x00711918 / 0x0)

The Event Queue Head Registers comprise 36 consecutive registers, one for each EQ. They are read/write registers for software which uses them to read and write the hardware value of the EQ head pointer.

Table 1-115 Event Queue Head Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:7				Reserved field
HEAD	6:0	rst_1	0x0	RW	EQ Head Pointer. Initialized by SW, written by SW during operation.

1.3.4.17 MSI Mapping Register (0x00620000-0x006207F8, 0x00720000-0x007207F8 / 0x0)

The MSI Mapping Registers comprise 256 consecutive registers, one for each MSI. They are read/write registers which software uses to set up each of the 256 supported MSI's. Through these registers, software, on a per-MSI basis, can enable and disable a given

MSI's valid bit as well as select which event queue the MSI will use. In addition, software can also obtain status on the value of the 'EQWR_N' bit which is used by hardware to determine if it is OK to send an MSI to the EQ or if it is a duplicate.

Table 1-116 MSI Mapping Register

Field	Bits	Reset Name	Reset Value	Type	Description
V	63	rst_1	0x0	RW	0 - Not Valid. A received MSI with this number will be treated as an error. 1 - Valid. A received MSI with this number will be routed to the EQ specified in the EQNUM field
EQWR_N	62	rst_1	0x0	R	0 - OK to write to. A received MSI with this number will be sent to the EQ specified. 1 - MSI already in EQ. A received MSI with this number will be treated as a duplicate. SW must clear this bit BEFORE calling the client's interrupt handler.
RESERVED	61:6				Reserved field
EQNUM	5:0	rst_1	0x0	RW	Event Queue Number

1.3.4.18 MSI Clear Registers (0x00628000-0x006287F8, 0x00728000-0x007287F8 / 0x0)

The MSI Clear Registers comprise 256 consecutive registers, one for each MSI. They are read/write-one-to-clear registers software uses to clear the EQWR_N bit for each of the supported 256 MSI's.

Table 1-117 MSI Clear Registers

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63				Reserved field
EQWR_N	62	rst_1	0x0	RW1C	Write 0 has no effect. Write 1 will clear the EQWR_N bit of the MSI Mapping Register. When reading, the value of the EQWR_N bit from the MSI Mapping Register will be returned.
RESERVED	61:0				Reserved field

1.3.4.19 Interrupt Mondo Data 0 Register (0x0062C000, 0x0072C000 / 0x0)

The Interrupt Mondo Data 0 Register is a read/write register software uses to set the value of the upper bits of Data Word 0 for a data bearing mondo. This value will be inserted into the upper bits of Data Word 0 when the mondo's mode is selected to be 1'b1 in the Interrupt Mapping Register

Table 1-118 Interrupt Mondo Data 0 Register

Field	Bits	Reset Name	Reset Value	Type	Description
DATA	63:6	rst_l	0x0	RW	Data 0 Word, bits 63:6 of mondo used for a data bearing mondo with the mode bit set to 1.
RESERVED	5:0				Reserved field

1.3.4.20 Interrupt Mondo Data 1 Register (0x0062C008, 0x0072C008 / 0x0)

The Interrupt Mondo Data 1 Register is a read/write register software uses to set the value of Data Word 1 for a data bearing mondo. This value will be inserted into Data Word 1 when the mondo's mode is selected to be 1'b1 in the Interrupt Mapping Register

Table 1-119 Interrupt Mondo Data 1 Register

Field	Bits	Reset Name	Reset Value	Type	Description
DATA	63:0	rst_l	0x0	RW	Data Word 1 of mondo used for data bearing mondos with the mode bit set to 1.

1.3.4.21 ERR COR Mapping Register (0x00630000, 0x00730000 / 0x0)

The ERR COR Mapping Register is a read/write register software uses to control how hardware handles the reception of an ERR COR message. Through this register, software enables or disables the ERR COR message's valid bit as well as selects which of the 36 EQ's the message will use. Please note that setting the EQNUM field to a value larger than 35 will yield unpredictable results.

Table 1-120 ERR COR Mapping Register

Field	Bits	Reset Name	Reset Value	Type	Description
V	63	rst_1	0x0	RW	0 - Not Valid. A received message of this type will be treated as an error. 1 - Valid. A received message of this type will be routed to the EQ specified in the EQNUM field
RESERVED	62:6				Reserved field
EQNUM	5:0	rst_1	0x0	RW	Event Queue Number

1.3.4.22 ERR NONFATAL Mapping Register (0x00630008, 0x00730008 / 0x0)

The ERR NONFATAL Mapping Register is a read/write register software uses to control how hardware handles the reception of an ERR NONFATAL message. Through this register, software enables or disables the ERR NONFATAL message's valid bit as well as selects which of the 36 EQ's the message will use. Please note that setting the EQNUM field to a value larger than 35 will yield unpredictable results.

Table 1-121 ERR NONFATAL Mapping Register

Field	Bits	Reset Name	Reset Value	Type	Description
V	63	rst_1	0x0	RW	0 - Not Valid. A received message of this type will be treated as an error. 1 - Valid. A received message of this type will be routed to the EQ specified in the EQNUM field
RESERVED	62:6				Reserved field
EQNUM	5:0	rst_1	0x0	RW	Event Queue Number

1.3.4.23 ERR FATAL Mapping Register (0x00630010, 0x00730010 / 0x0)

The ERR FATAL Mapping Register is a read/write register software uses to control how hardware handles the reception of an ERR FATAL message. Through this register software enables or disables the ERR FATAL message's valid bit as well as selects which of the 36 EQ's the message will use. Please note that setting the EQNUM field to a value larger than 35 will yield unpredictable results.

Table 1-122 ERR FATAL Mapping Register

Field	Bits	Reset Name	Reset Value	Type	Description
V	63	rst_1	0x0	RW	0 - Not Valid. A received message of this type will be treated as an error. 1 - Valid. A received message of this type will be routed to the EQ specified in the EQNUM field
RESERVED	62:6				Reserved field
EQNUM	5:0	rst_1	0x0	RW	Event Queue Number

1.3.4.24 PM PME Mapping Register (0x00630018, 0x00730018 / 0x0)

The PM PME Mapping Register is a read/write register software uses to control how hardware should handle the reception of a PM PME message. Through this register software enables or disables the PM PME message's valid bit as well as selects which of the 36 EQ's the message will use. Please note that setting the EQNUM field to a value larger than 35 will yield unpredictable results.

Table 1-123 PM PME Mapping Register

Field	Bits	Reset Name	Reset Value	Type	Description
V	63	rst_1	0x0	RW	0 - Not Valid. A received message of this type will be treated as an error. 1 - Valid. A received message of this type will be routed to the EQ specified in the EQNUM field
RESERVED	62:6				Reserved field
EQNUM	5:0	rst_1	0x0	RW	Event Queue Number

1.3.4.25 PME To ACK Mapping Register (0x00630020, 0x00730020 / 0x0)

The PME To ACK Mapping Register is a read/write register software uses to control how hardware handles the reception of a PME To ACK message. Through this register software enables or disables the PME To ACK message's valid bit as well as selects which of the 36 EQ's the message will use. Please note that setting the EQNUM field to a value larger than 35 will yield unpredictable results.

Table 1-124 PME To ACK Mapping Register

Field	Bits	Reset Name	Reset Value	Type	Description
V	63	rst_1	0x0	RW	0 - Not Valid. A received message of this type will be treated as an error. 1 - Valid. A received message of this type will be routed to the EQ specified in the EQNUM field
RESERVED	62:6				Reserved field
EQNUM	5:0	rst_1	0x0	RW	Event Queue Number

1.3.4.26 IMU Error Log Enable Register (0x00631000, 0x00731000 / 0x7FFF)

The IMU Error Log Enable Register is used to enable logging for all of the possible IMU errors detected by Fire. By default all of the possible errors are enabled for logging.

The IMU supports three groupings of errors: SCS, EQS, and RDS. The errors associated with each group are as follows:

RDS Group: msi_mal_err, msi_par_err, pmeack_mes_not_en, pmpme_mes_not_en, fatal_mes_not_en, nonfatal_mes_not_en, cor_mes_not_en, msi_not_en, spare bit 0, spare bit 1

SCS Group: eq_not_en

EQS Group: eq_over

Please note that since the RDS sub-block can detect multiple possible errors for a MSI transaction, there is a priority order for the MSI errors in the RDS group. Of the 3 detected MSI errors listed above, the priority order will be: msi_par_err, msi_mal_err, msi_not_en. The EQWR_N bit will not be affected by any of these three errors and will remain unchanged, holding the value from the last non-errored MSI.

Table 1-125 IMU Error Log Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:15				Reserved field

Table 1-125 IMU Error Log Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
SPARE_LOG_EN	14:10	por_1	0x1F	RW	Spare Error, Error Log Enable Bits
EQ_OVER_LOG_EN	9	por_1	0x1	RW	EQ Overflow Error, Error Log Enable Bit
EQ_NOT_EN_LOG_EN	8	por_1	0x1	RW	EQ Not Enabled, Error Log Enable Bit
MSI_MAL_ERR_LOG_EN	7	por_1	0x1	RW	Malformed MSI, Error Log Enable Bit
MSI_PAR_ERR_LOG_EN	6	por_1	0x1	RW	MSI Data Parity Error, Error Log Enable Bit
PMEACK_MES_NOT_EN_LOG_EN	5	por_1	0x1	RW	PME to ACK Message Not Enabled, Error Log Enable Bit
PMPME_MES_NOT_EN_LOG_EN	4	por_1	0x1	RW	PM PME Message Not Enabled, Error Log Enable Bit
FATAL_MES_NOT_EN_LOG_EN	3	por_1	0x1	RW	Fatal Message Not Enabled, Error Log Enable Bit
NONFATAL_MES_NOT_EN_LOG_EN	2	por_1	0x1	RW	Non Fatal Message Not Enabled, Error Log Enable Bit
COR_MES_NOT_EN_LOG_EN	1	por_1	0x1	RW	Correctable Message Not Enabled, Error Log Enable Bit
MSI_NOT_EN_LOG_EN	0	por_1	0x1	RW	MSI Not Enabled, Error Log Enable Bit

1.3.4.27 IMU Interrupt Enable Register (0x00631008, 0x00731008 / 0x0)

The IMU Error Log Enable Register is used to enable interrupts for all possible IMU errors detected by Fire. By default interrupt generation for all of these errors is disabled.

Table 1-126 IMU Interrupt Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:47				Reserved field

Table 1-126 IMU Interrupt Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
SPARE_S_INT_EN	46:42	rst_1	0x0	RW	Spare Error, Secondary Interrupt Enable Bits
EQ_OVER_S_INT_EN	41	rst_1	0x0	RW	EQ Overflow Error, Secondary Interrupt Enable Bit
EQ_NOT_EN_S_INT_EN	40	rst_1	0x0	RW	EQ Not Enabled, Secondary Interrupt Enable Bit
MSI_MAL_ERR_S_INT_EN	39	rst_1	0x0	RW	Malformed MSI, Secondary Interrupt Enable Bit
MSI_PAR_ERR_S_INT_EN	38	rst_1	0x0	RW	MSI Data Parity Error, Secondary Interrupt Enable Bit
PMEACK_MES_NOT_EN_S_INT_EN	37	rst_1	0x0	RW	PME to ACK Message Not Enabled, Secondary Interrupt Enable Bit
PMPME_MES_NOT_EN_S_INT_EN	36	rst_1	0x0	RW	PM PME Message Not Enabled, Secondary Interrupt Enable Bit
FATAL_MES_NOT_EN_S_INT_EN	35	rst_1	0x0	RW	Fatal Message Not Enabled, Secondary Interrupt Enable Bit
NONFATAL_MES_NOT_EN_S_INT_EN	34	rst_1	0x0	RW	Non Fatal Message Not Enabled, Secondary Interrupt Enable Bit
COR_MES_NOT_EN_S_INT_EN	33	rst_1	0x0	RW	Correctable Message Not Enabled, Secondary Interrupt Enable Bit
MSI_NOT_EN_S_INT_EN	32	rst_1	0x0	RW	MSI Not Enabled, Secondary Interrupt Enable Bit
RESERVED	31:15				Reserved field
SPARE_P_INT_EN	14:10	rst_1	0x0	RW	Spare Error, Primary Interrupt Enable Bits
EQ_OVER_P_INT_EN	9	rst_1	0x0	RW	EQ Overflow Error, Primary Interrupt Enable Bit
EQ_NOT_EN_P_INT_EN	8	rst_1	0x0	RW	EQ Not Enabled, Primary Interrupt Enable Bit

Table 1-126 IMU Interrupt Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
MSI_MAL_ERR_P_INT_EN	7	rst_1	0x0	RW	Malformed MSI, Primary Interrupt Enable Bit
MSI_PAR_ERR_P_INT_EN	6	rst_1	0x0	RW	MSI Data Parity Error, Primary Interrupt Enable Bit
PMEACK_MES_NOT_EN_P_INT_EN	5	rst_1	0x0	RW	PME to ACK Message Not Enabled, Primary Interrupt Enable Bit
PMPME_MES_NOT_EN_P_INT_EN	4	rst_1	0x0	RW	PM PME Message Not Enabled, Primary Interrupt Enable Bit
FATAL_MES_NOT_EN_P_INT_EN	3	rst_1	0x0	RW	Fatal Message Not Enabled, Primary Interrupt Enable Bit
NONFATAL_MES_NOT_EN_P_INT_EN	2	rst_1	0x0	RW	Non Fatal Message Not Enabled, Primary Interrupt Enable Bit
COR_MES_NOT_EN_P_INT_EN	1	rst_1	0x0	RW	Correctable Message Not Enabled, Primary Interrupt Enable Bit
MSI_NOT_EN_P_INT_EN	0	rst_1	0x0	RW	MSI Not Enabled, Primary Interrupt Enable Bit

1.3.4.28 IMU Interrupt Status Register (0x00631010, 0x00731010 / 0x0)

The IMU Interrupt Status Register is a read-only register software reads to obtain the source of interrupts for the IMU Block. (Mondo number 62). This register will contain a 1 in any and all bits where an interrupt was generated.

Table 1-127 IMU Interrupt Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:47				Reserved field
SPARE_S	46:42	rst_1	0x0	R	Spare Error, Secondary Error Status Bit (1 = Error Received)
EQ_OVER_S	41	rst_1	0x0	R	EQ Overflow Secondary Error Status Bit (1 = Error Received)
EQ_NOT_EN_S	40	rst_1	0x0	R	EQ Not Enabled Secondary Error Status Bit (1 = Error Received)

Table 1-127 IMU Interrupt Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
MSI_MAL_ERR_S	39	rst_1	0x0	R	Malformed MSI Secondary Error Status Bit (1 = Error Received)
MSI_PAR_ERR_S	38	rst_1	0x0	R	MSI Data Parity Secondary Error Status Bit (1 = Error Received)
PMEACK_MES_NOT_EN_S	37	rst_1	0x0	R	PME to ACK Message Not Enabled Secondary Error Status Bit (1 = Error Received)
PMPME_MES_NOT_EN_S	36	rst_1	0x0	R	PM PME Message Not Enabled Secondary Error Status Bit (1 = Error Received)
FATAL_MES_NOT_EN_S	35	rst_1	0x0	R	Fatal Message Not Enabled Secondary Error Status Bit (1 = Error Received)
NONFATAL_MES_NOT_EN_S	34	rst_1	0x0	R	Non Fatal Message Not Enabled Secondary Error Status Bit (1 = Error Received)
COR_MES_NOT_EN_S	33	rst_1	0x0	R	Correctable Message Not Enabled Secondary Error Status Bit (1 = Error Received)
MSI_NOT_EN_S	32	rst_1	0x0	R	MSI Not Enabled Secondary Error Status Bit (1 = Error Received)
RESERVED	31:15				Reserved field
SPARE_P	14:10	rst_1	0x0	R	Spare Error, Primary Error Status Bit (1 = Error Received)
EQ_OVER_P	9	rst_1	0x0	R	EQ Overflow Primary Error Status Bit (1 = Error Received)
EQ_NOT_EN_P	8	rst_1	0x0	R	EQ Not Enabled Primary Error Status Bit (1 = Error Received)
MSI_MAL_ERR_P	7	rst_1	0x0	R	Malformed MSI Primary Error Status Bit (1 = Error Received)
MSI_PAR_ERR_P	6	rst_1	0x0	R	MSI Data Parity Primary Error Status Bit (1 = Error Received)

Table 1-127 IMU Interrupt Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
PMEACK_MES_NOT_EN_P	5	rst_1	0x0	R	PME to ACK Message Not Enabled Primary Error Status Bit (1 = Error Received)
PMPME_MES_NOT_EN_P	4	rst_1	0x0	R	PM PME Message Not Enabled Primary Error Status Bit (1 = Error Received)
FATAL_MES_NOT_EN_P	3	rst_1	0x0	R	Fatal Message Not Enabled Primary Error Status Bit (1 = Error Received)
NONFATAL_MES_NOT_EN_P	2	rst_1	0x0	R	Non Fatal Message Not Enabled Primary Error Status Bit (1 = Error Received)
COR_MES_NOT_EN_P	1	rst_1	0x0	R	Correctable Message Not Enabled Primary Error Status Bit (1 = Error Received)
MSI_NOT_EN_P	0	rst_1	0x0	R	MSI Not Enabled Primary Error Status Bit (1 = Error Received)

1.3.4.29 IMU Error Status Clear Register (0x00631018, 0x00731018/0x0)

The IMU Error Status Clear Register is a read/write-one-to-clear register serving two purposes for software. First, software can read this register to determine which IMU errors have been logged by Fire but have not necessarily caused an interrupt (Mondo 62). For all logged errors, this register will contain a one in the associated bit. Second, software uses this register to clear the IMU errors detected by Fire by writing a one to the bit(s) which it wants to clear.

Table 1-128 IMU Error Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:47				Reserved field
SPARE_S	46:42	por_1	0x0	RW1C	Spare Error, Secondary Error Status Bit (1 = Error Received)
EQ_OVER_S	41	por_1	0x0	RW1C	EQ Overflow Secondary Error Status Bit (1 = Error Received)

Table 1-128 IMU Error Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
EQ_NOT_EN_S	40	por_1	0x0	RW1C	EQ Not Enabled Secondary Error Status Bit (1 = Error Received)
MSI_MAL_ERR_S	39	por_1	0x0	RW1C	Malformed MSI Secondary Error Status Bit (1 = Error Received)
MSI_PAR_ERR_S	38	por_1	0x0	RW1C	MSI Data Parity Secondary Error Status Bit (1 = Error Received)
PMEACK_MES_NOT_EN_S	37	por_1	0x0	RW1C	PME to ACK Message Not Enabled Primary Error Status Bit (1 = Error Received)
PMPME_MES_NOT_EN_S	36	por_1	0x0	RW1C	PM PME Message Not Enabled Primary Error Status Bit (1 = Error Received)
FATAL_MES_NOT_EN_S	35	por_1	0x0	RW1C	Fatal Message Not Enabled Primary Error Status Bit (1 = Error Received)
NONFATAL_MES_NOT_EN_S	34	por_1	0x0	RW1C	Non Fatal Message Not Enabled Primary Error Status Bit (1 = Error Received)
COR_MES_NOT_EN_S	33	por_1	0x0	RW1C	Correctable Message Not Enabled Primary Error Status Bit (1 = Error Received)
MSI_NOT_EN_S	32	por_1	0x0	RW1C	MSI Not Enabled Secondary Error Status Bit (1 = Error Received)
RESERVED	31:15				Reserved field
SPARE_P	14:10	por_1	0x0	RW1C	Spare Error, Primary Error Status Bit (1 = Error Received)
EQ_OVER_P	9	por_1	0x0	RW1C	EQ Overflow Primary Error Status Bit (1 = Error Received)
EQ_NOT_EN_P	8	por_1	0x0	RW1C	EQ Not Enabled Primary Error Status Bit (1 = Error Received)
MSI_MAL_ERR_P	7	por_1	0x0	RW1C	Malformed MSI Primary Error Status Bit (1 = Error Received)

Table 1-128 IMU Error Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
MSI_PAR_ERR_P	6	por_1	0x0	RW1C	MSI Data Parity Primary Error Status Bit (1 = Error Received)
PMEACK_MES_NOT_EN_P	5	por_1	0x0	RW1C	PME to ACK Message Not Enabled Primary Error Status Bit (1 = Error Received)
PMPME_MES_NOT_EN_P	4	por_1	0x0	RW1C	PM PME Message Not Enabled Primary Error Status Bit (1 = Error Received)
FATAL_MES_NOT_EN_P	3	por_1	0x0	RW1C	Fatal Message Not Enabled Primary Error Status Bit (1 = Error Received)
NONFATAL_MES_NOT_EN_P	2	por_1	0x0	RW1C	Non Fatal Message Not Enabled Primary Error Status Bit (1 = Error Received)
COR_MES_NOT_EN_P	1	por_1	0x0	RW1C	Correctable Message Not Enabled Primary Error Status Bit (1 = Error Received)
MSI_NOT_EN_P	0	por_1	0x0	RW1C	MSI Not Enabled Primary Error Status Bit (1 = Error Received)

1.3.4.30 IMU Error Status Set Register (0x00631020, 0x00731020 / 0x0)

The IMU Error Status Set Register is a read/write-one-to-set register serving two purposes for software. First, software can read this register to determine which IMU errors have been logged by Fire but have not necessarily caused an interrupt (Mondo 62). For all logged errors, this register will contain a one in the associated bit. Second, software can use this register to set the IMU errors detected by Fire by writing a one to the bit(s) which it wants to set. THIS SHOULD ONLY BE USED FOR DIAG PURPOSES AND ERROR TESTING.

Table 1-129 IMU Error Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:47				Reserved field
SPARE_S	46:42	por_1	0x0	RW1S	Spare Error, Secondary Error Status Bit (1 = Error Received)

Table 1-129 IMU Error Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
EQ_OVER_S	41	por_1	0x0	RW1S	EQ Overflow Secondary Error Status Bit (1 = Error Received)
EQ_NOT_EN_S	40	por_1	0x0	RW1S	EQ Not Enabled Secondary Error Status Bit (1 = Error Received)
MSI_MAL_ERR_S	39	por_1	0x0	RW1S	Malformed MSI Secondary Error Status Bit (1 = Error Received)
MSI_PAR_ERR_S	38	por_1	0x0	RW1S	MSI Data Parity Secondary Error Status Bit (1 = Error Received)
PMEACK_MES_NOT_EN_S	37	por_1	0x0	RW1S	PME to ACK Message Not Enabled Primary Error Status Bit (1 = Error Received)
PMPME_MES_NOT_EN_S	36	por_1	0x0	RW1S	PM PME Message Not Enabled Primary Error Status Bit (1 = Error Received)
FATAL_MES_NOT_EN_S	35	por_1	0x0	RW1S	Fatal Message Not Enabled Primary Error Status Bit (1 = Error Received)
NONFATAL_MES_NOT_EN_S	34	por_1	0x0	RW1S	Non Fatal Message Not Enabled Primary Error Status Bit (1 = Error Received)
COR_MES_NOT_EN_S	33	por_1	0x0	RW1S	Correctable Message Not Enabled Primary Error Status Bit (1 = Error Received)
MSI_NOT_EN_S	32	por_1	0x0	RW1S	MSI Not Enabled Secondary Error Status Bit (1 = Error Received)
RESERVED	31:15				Reserved field
SPARE_P	14:10	por_1	0x0	RW1S	Spare Error, Primary Error Status Bit (1 = Error Received)
EQ_OVER_P	9	por_1	0x0	RW1S	EQ Overflow Primary Error Status Bit (1 = Error Received)
EQ_NOT_EN_P	8	por_1	0x0	RW1S	EQ Not Enabled Primary Error Status Bit (1 = Error Received)

Table 1-129 IMU Error Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
MSI_MAL_ERR_P	7	por_1	0x0	RW1S	Malformed MSI Primary Error Status Bit (1 = Error Received)
MSI_PAR_ERR_P	6	por_1	0x0	RW1S	MSI Data Parity Primary Error Status Bit (1 = Error Received)
PMEACK_MES_NOT_EN_P	5	por_1	0x0	RW1S	PME to ACK Message Not Enabled Primary Error Status Bit (1 = Error Received)
PMPME_MES_NOT_EN_P	4	por_1	0x0	RW1S	PM PME Message Not Enabled Primary Error Status Bit (1 = Error Received)
FATAL_MES_NOT_EN_P	3	por_1	0x0	RW1S	Fatal Message Not Enabled Primary Error Status Bit (1 = Error Received)
NONFATAL_MES_NOT_EN_P	2	por_1	0x0	RW1S	Non Fatal Message Not Enabled Primary Error Status Bit (1 = Error Received)
COR_MES_NOT_EN_P	1	por_1	0x0	RW1S	Correctable Message Not Enabled Primary Error Status Bit (1 = Error Received)
MSI_NOT_EN_P	0	por_1	0x0	RW1S	MSI Not Enabled Primary Error Status Bit (1 = Error Received)

1.3.4.31 IMU RDS Error Log Register (0x00631028, 0x00731028 / 0x0)

The IMU RDS Error Log Register is used to capture information when the first primary error detected by the IMU is in the RDS group. Please note that until an error in the IMU RDS Group is logged, this register does not contain valid data and will update on every cycle. The RDS Error Log Register only contains the first two bytes of MSI data. See the bit descriptions below for order details.

The RDS error logging register will NOT change for the MSI-X RFE and WILL NOT have all 4 bytes of the MSI data stored in it. It will only contain the first two bytes of MSI DATA.

Table 1-130 IMU RDS Error Log Register

Field	Bits	Reset Name	Reset Value	Type	Description
TYPE	63:58	por_1	6'bx	RW	The lowest 6 bits of the Type of the errored transaction as seen by the IMU in the RDS pipe stage. 1111000 - 64 bit addressed MSI 1011000 - 32 bit addressed MSI 0110xxx - Message where xxx complies with the routing code as specified by PCI-E.
LENGTH	57:48	por_1	10'bx	RW	The Length of the errored transaction
REQ_ID	47:32	por_1	16'bx	RW	The REQ ID of the errored transaction
TLP_TAG	31:24	por_1	8'bx	RW	The TLP tag of the errored transaction
BE_MESS_CODE	23:16	por_1	8'bx	RW	The message code if the error is associated with a message; the first and last byte enables if the error is associated with an MSI.
MSI_DATA	15:0	por_1	16'bx	RW	The first two bytes MSI data if the error is associated with a MSI (byte 1, byte 0)

1.3.4.32 IMU SCS Error Log Register (0x00631030, 0x00731030 / 0x0)

The IMU SCS Error Log Register is used to capture information when the first primary error detected by the IMU is in the SCS group. Please note that until an error in the IMU SCS Group is logged, this register does not contain valid data and will update on every cycle.

Table 1-131 IMU SCS Error Log Register

Field	Bits	Reset Name	Reset Value	Type	Description
TYPE	63:58	por_1	6'bx	RW	The lowest 6 bits of the type of the errored transaction as seen by the IMU in the SCS pipe stage: 1011000 = MSI 1010000 = Message
LENGTH	57:48	por_1	10'bx	RW	The Length of the errored transaction. At this stage of the IMU pipeline the length has already been fixed so this always will be 10'h10
REQ_ID	47:32	por_1	16'bx	RW	The REQ ID of the errored transaction.
TLP_TAG	31:24	por_1	8'bx	RW	The TLP tag of the errored transaction.
BE_MESS_CODE	23:16	por_1	8'bx	RW	The message code if the error is associated with a message; the first and last byte enables if the error is associated with an MSI.
RESERVED	15:6				Reserved field
EQ_NUM	5:0	por_1	6'bx	RW	EQ Number that the transaction attempted to use.

1.3.4.33 IMU EQS Error Log Register (0x00631038, 0x00731038 / 0x0)

The IMU EQS Error Log Register is used to capture information when the first primary error detected by the IMU is in the EQS group. Please note that until an error in the IMU EQS Group is logged, this register does not contain valid data and will be updated on every cycle.

Table 1-132 IMU EQS Error Log Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:6				Reserved field
EQ_NUM	5:0	por_1	6'bx	RW	EQ number that the transaction attempted to use.

1.3.4.34 DMC Core and Block Interrupt Enable Register (0x00631800, 0x00731800 / 0x0)

The DMC Core and Block Interrupt Enable Register is used by software to mask interrupts it does not want to receive; it can mask at core and block level granularity. Each of the supported DMC errors falls under one of the block enables (MMU or IMU). If a particular block is not enabled, no errors from that block will generate an interrupt (regardless of the individual error's interrupt enable setting). If the DMC core interrupt enable is not enabled no JBC errors will generate interrupts.

Please note that if a condition occurs that should have caused an interrupt via mondo 62 but a particular error-enable is disabled at that time (thus the mondo is not sent), then if that error-enable is enabled before clearing the stored offending condition, an interrupt via mondo 62 will be sent.

Table 1-133 DMC Core and Block Interrupt Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
DMC	63	rst_1	0x0	RW	The enable bit to enable all operations from the DMC which would cause an interrupt via mondo 62: 1 = core level interrupt enabled, 0 = core level interrupt disabled.
RESERVED	62:2				Reserved field
MMU	1	rst_1	0x0	RW	The enable bit to enable all operations from the MMU which would cause an interrupt via mondo 62: 1 = block level interrupt enabled, 0 = block level interrupt disabled.

Table 1-133 DMC Core and Block Interrupt Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
IMU	0	rst_l	0x0	RW	The enable bit to enable all operations from the IMU which would cause an interrupt via mondo 62: 1 = block level interrupt enabled, 0 = block level interrupt disabled.

1.3.4.35 DMC Core and Block Error Status Register (0x00631808, 0x00731808 / 0x0)

The DMC Core and Block Error Status Register is a read-only register software uses to determine which block in the DMC core caused an interrupt when software receives a mondo 62 from the DMC core. A value of one means that particular block caused an interrupt to be generated

Table 1-134 DMC Core and Block Error Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:2				Reserved field
MMU	1	rst_l	0x0	R	MMU has an interrupt that needs to be processed via mondo 62
IMU	0	rst_l	0x0	R	IMU has an interrupt that needs to be processed via mondo 62

1.3.4.36 Multi Core Error Status Register (0x00631810, 0x00731810 / 0x0)

The Multi Core and Block Error Status Register is a read-only register software uses to determine which core in Fire, the DMC or PEC, caused an interrupt when software receives a mondo 62. A value of 1 means that particular core caused an interrupt to be generated

Table 1-135 Multi Core Error Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:2				Reserved field
PEC	1	rst_l	0x0	R	PEC has an interrupt that needs to be processed via mondo 62
DMC	0	rst_l	0x0	R	DMC has an interrupt that needs to be processed via mondo 62

1.3.4.37 IMU Performance Counter Select Register (0x00632000, 0x00732000 / 0x0)

The IMU Performance Counter Select Register provides the select inputs for both of the performance counters in the IMU block. These selects are used to select which of seven possible events each counter should count. The counters both default to disabled. When selecting an event to count, the counting begins as soon as this register is updated. It should be noted that by changing the value of the select, the counter register DOES NOT reset to 0. It will continue to count at its current value. If the counter needs to start at 0, it should be cleared before the new value of the select is written

Table 1-136 IMU Performance Counter Select Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:16				Reserved field
SEL1	15:8	rst_1	0x0	RW	The eligible values for SEL1 are the same as SEL0
SEL0	7:0	rst_1	0x0	RW	Select for counter 0: 00 = None 01 = Clock cycles 02 = Total Number of Mondo's Issued 03 = Total Number of MSI's Issued 04 = Total Number of Mondo NACK's 05 = Total Number of EQ Writes 06 = Total Number of EQ Mondo's

1.3.4.38 IMU Performance Counter Zero Register (0x00632008, 0x00732008 / 0x0)

The IMU Performance Counter Zero Register is used to read and write the value of Counter Zero for the IMU block. This counter will increment once for every clock cycle that the selected event chosen by the IMU Performance Counter Select Register occurs. This value can be set and cleared by software.

Table 1-137 IMU Performance Counter Zero Register

Field	Bits	Reset Name	Reset Value	Type	Description
CNT	63:0	rst_1	0x0	RW	Counter

1.3.4.39 IMU Performance Counter One Register (0x00632010, 0x00732010 / 0x0)

The IMU Performance Counter One Register is used to read and write the value of Counter one for the IMU block. This counter will increment once for every clock cycle that the selected event chosen by the IMU Performance Counter Select Register occurs. This value can be set and cleared by software.

Table 1-138 IMU Performance Counter One Register

Field	Bits	Reset Name	Reset Value	Type	Description
CNT	63:0	rst_l	0x0	RW	Counter

1.3.4.40 MSI 32-bit Address Register (0x00634000, 0x00734000 / 0x0)

The MSI 32-bit Address Register is used by hardware to compare against the address of incoming PCI-E 32-bit addressed memory write commands. If the address matches bits 31:16 the PCI-E command is considered to be an MSI and is thus passed to the IMU to be processed. The value of this register is programmed by software.

Table 1-139 MSI 32-bit Address Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
ADDR	31:16	rst_l	0x0	RW	Upper 16 bits of the 32-bit address translated as a MSI Interrupt.
RESERVED	15:0				Reserved field

1.3.4.41 MSI 64-bit Address Register (0x00634008, 0x00734008 / 0x0)

The MSI 64-bit Address Register is used by hardware to compare against the address of incoming PCI-E 64-bit addressed memory write commands. If the address matches bits 63:16 the PCI-E command is considered to be an MSI and is thus passed to the IMU to be processed. The value of this register is programmed by software.

Table 1-140 MSI 64-bit Address Register

Field	Bits	Reset Name	Reset Value	Type	Description
ADDR	63:16	rst_l	0x0	RW	Upper 48 bits of the 64-bit address translated as a MSI Interrupt.
RESERVED	15:0				Reserved field

1.3.4.42 Mem 64 PCIE Offset Register (0x00634018, 0x00734018 / 0x0)

The Mem 64 PCIE Offset Register serves two purposes. First, the register is written by software to select the value of the upper bits of the address for a 64 bit addressed PIO initiated by FIRE which hardware will place in the PCI-E command header before the packet is sent. Second, the register contains 24 spare control and status bits.-which could be used in a metal spin. These bits currently have no functionality.

Table 1-141 Mem 64 PCIE Offset Register

Field	Bits	Reset Name	Reset Value	Type	Description
ADDR	63:24	rst_1	0x0	RW	The upper 40 bits of the 64 bit PIO address. The 64-bit PIO address is formed from {offset[63:36], (offset[35:24] jbus addr[35:24]), jbus addr[23:2]} where jbus addr[35:0] = (PA[35:24] & ~mask[35:24]), PA[23:0]
SPARE_CONTROL_LOAD_7	23	rst_1	0x0	RW	Spare Loadable Control Register 7
SPARE_CONTROL_LOAD_6	22	rst_1	0x0	RW	Spare Loadable Control Register 6
SPARE_CONTROL_LOAD_5	21	rst_1	0x0	RW	Spare Loadable Control Register 5
SPARE_CONTROL_LOAD_4	20	rst_1	0x0	RW	Spare Loadable Control Register 4
SPARE_CONTROL_LOAD_3	19	rst_1	0x0	RW	Spare Loadable Control Register 3
SPARE_CONTROL_LOAD_2	18	rst_1	0x0	RW	Spare Loadable Control Register 2
SPARE_CONTROL_LOAD_1	17	rst_1	0x0	RW	Spare Loadable Control Register 1
SPARE_CONTROL_LOAD_0	16	rst_1	0x0	RW	Spare Loadable Control Register 0
SPARE_CONTROL	15:8	rst_1	0x0	RW	Spare Control Registers
SPARE_STATUS	7:0	rst_1	0x0	RW	Spare Status Registers

1.3.4.43 MMU Control and Status Register (0x00640000, 0x00740000 / 0x0)

The MMU Control and Status Register enables the functions of the MMU and indicates MMU status.

Table 1-142 MMU Control and Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:52				Reserved field
SPARES	51:48	rst_1	0x0	R	Spare status bits.
RESERVED	47:46				Reserved field
PAQ	45	rst_1	0x0	R	Physical Address Queue not empty.
VAQ	44	rst_1	0x0	R	Virtual Address Queue not empty.
TPL	43	rst_1	0x0	R	Translation Pipeline not empty.
TIP	42	rst_1	0x0	R	Tablewalk in Progress.
TCM	41:40	rst_1	0x0	R	Tablewalk Cache Mode. Cache mode when tablewalk was initiated.
RESERVED	39:20				Reserved field
SPAREC	19:16	rst_1	0x0	RW	Spare control bits.
RESERVED	15:13				Reserved field
PD	12	rst_1	0x0	RW	Process Disable. When set, addresses are not processed. Otherwise, addresses are processed. During normal operation, this bit will be clear.
RESERVED	11				Reserved field
SE	10	rst_1	0x0	RW	TSB Cache Snoop Enable. When set, snoop addresses which hit will invalidate cache entries and/or the TLB. This is the enable for the TSB record. During normal operation, this bit will be set.

Table 1-142 MMU Control and Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
CM	9:8	rst_l	0x0	RW	Cache Mode. 00 = Cache disabled, TLB used, tablewalks enabled. 01 = Cache enabled but locked, tablewalks disabled. 10 = Cache enabled but locked, TLB used, tablewalks enabled. 11 = Cache enabled, TLB used for update, tablewalks enabled. During normal operation, it's value is 11.
RESERVED	7:2				Reserved field
BE	1	rst_l	0x0	RW	Bypass Enable. When set, bypass mode is enabled. Otherwise, an attempt to use bypass mode causes an error. During normal operation, this bit will be set.
TE	0	rst_l	0x0	RW	Translation Enable. When set, translation mode is enabled. Otherwise, an attempt to use translation mode causes an error. During normal operation, this bit will be set.

1.3.4.44 MMU TSB Control Register (0x00640008, 0x00740008 / 0x0)

The TSB Control Register contains the pointer to the first-entry of the TSB table. Together with part of the virtual address it uniquely identifies the address where hardware should fetch the TTE from the TSB table. The TSB table has to be aligned on an 8K boundary. The lower order 13 address bits are assumed to be 0x0 during TSB table lookup. Tables larger than 8K bytes are only constrained to be on 8K boundaries rather than having to be size aligned. The TSB size and TSB page size are also configured in this register.

Table 1-143 MMU TSB Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:43				Reserved field
TB	42:13	rst_l	0x0	RW	Base address. Most significant 30 bits of the TSB physical address.

Table 1-143 MMU TSB Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	12:9				Reserved field
PS	8	rst_l	0x0	RW	Page size (0 = 8 KB, 1 = 64 KB)
RESERVED	7:4				Reserved field
TS	3:0	rst_l	0x0	RW	TSB table size measured in the number of 8 byte entries. 0 = 1K, 1 = 2K, 2 = 4K, 3 = 8K, 4 = 16K, 5 = 32K, 6 = 64K, 7 = 128K, 8 = 256K, 9 = 512K, A-F = RESERVED

1.3.4.45 MMU TTE Cache Flush Address Register (0x00640100, 0x00740100 / 0x0)

The MMU TTE Cache Flush Address Register flushes an entry from the TTE cache if the address matches the physical address of the TSB entry corresponding to the cacheline to be flushed. The TLB is flushed if this address matches its physical tag. Writes are actually implemented outside of the MMU and use the hardware coherent flush interface. Reads which are not needed always return zero and are implemented here for software compatibility.

Note that this Register DOES NOT perform the same operation when written via JTAG. If a write to the register is done via JTAG, NO ENTRIES in the MMU will be flushed. This register will ONLY perform the flush operation from a CPU access. By writing this register via JTAG, the only effect will be to change the read return value for both access types to this address. This does not have an effect on normal operation. Software should ignore the read return value from this register as it does not provide any useful information in either access type. Software should never have a need to read this register.

Table 1-144 MMU TTE Cache Flush Address Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:43				Reserved field
FLSH_ADDR	42:6	rst_l	0x0	RW	Flush address

Table 1-144 MMU TTE Cache Flush Address Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	5:0				Reserved field

1.3.4.46 MMU TTE Cache Invalidate Register (0x00640108, 0x00740108 / 0x0)

The MMU TTE Cache Invalidate Register flushes the indicated entry from the TTE cache. Bit 0 flushes entry 0, etc. Writing all 1's will flush the entire cache. Any write will also flush the TLB. Writes to this register are not dependent upon the state of the SE or CM bits in the MMU Control and Status Register.

Table 1-145 MMU TTE Cache Invalidate Register

Field	Bits	Reset Name	Reset Value	Type	Description
FLSH_TTE	63:0	rst_1	0x0	L	Flush TTE cache entry

1.3.4.47 MMU Error Log Enable Register (0x00641000, 0x00741000 / 0xFFFF)

The MMU Error Log Enable Register enables the logging of errors.

Table 1-146 MMU Error Log Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:16				Reserved field
EN	15:0	por_1	0xFFFF	RW	Error log enables

1.3.4.48 MMU Interrupt Enable Register (0x00641008, 0x00741008 / 0x0)

The MMU Interrupt Enable Register enables errors to generate interrupts.

Table 1-147 MMU Interrupt Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:48				Reserved field
EN_S	47:32	rst_1	0x0	RW	Secondary interrupt enables
RESERVED	31:16				Reserved field
EN_P	15:0	rst_1	0x0	RW	Primary interrupt enables

1.3.4.49 MMU Interrupt Status Register (0x00641010, 0x00741010 / 0x0)

The MMU Interrupt Status Register shows the errors that are currently logged and enabled to generate interrupts.

Table 1-148 MMU Interrupt Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:48				Reserved field
ERR_S	47:32	rst_l	0x0	R	Secondary interrupt status
RESERVED	31:16				Reserved field
ERR_P	15:0	rst_l	0x0	R	Primary interrupt status

1.3.4.50 MMU Error Status Clear Register (0x00641018, 0x00741018 / 0x0)

The MMU Error Status Clear Register shows the errors that are currently logged. Primary errors include data logged with those errors. Secondary errors do not have any data logged with them. All errors belong to the same error group which means that any primary error logged will cause any additional error to get logged as secondary.

Table 1-149 MMU Error Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:48				Reserved field
TBW_DPE_S	47	por_l	0x0	RW1C	Tablewalk data parity secondary error
TBW_ERR_S	46	por_l	0x0	RW1C	Tablewalk secondary error
TBW_UDE_S	45	por_l	0x0	RW1C	Tablewalk unexpected data return secondary error
TBW_DME_S	44	por_l	0x0	RW1C	Tablewalk disabled miss secondary error
SPARE3_S	43	por_l	0x0	RW1C	Spare secondary error
SPARE2_S	42	por_l	0x0	RW1C	Spare secondary error
TTC_CAE_S	41	por_l	0x0	RW1C	TTE cache CSR access secondary error
TTC_DPE_S	40	por_l	0x0	RW1C	TTE cache data parity secondary error

Table 1-149 MMU Error Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
TTE_PRT_S	39	por_1	0x0	RW1C	TTE write bit not set on write secondary error
TTE_INV_S	38	por_1	0x0	RW1C	TTE valid bit not set secondary error
TRN_OOR_S	37	por_1	0x0	RW1C	Translation access out-of-range secondary error
TRN_ERR_S	36	por_1	0x0	RW1C	Translation access when TE = 0 secondary error
SPARE1_S	35	por_1	0x0	RW1C	Spare secondary error
SPARE0_S	34	por_1	0x0	RW1C	Spare secondary error
BYP_OOR_S	33	por_1	0x0	RW1C	Bypass access out-of-range secondary error
BYP_ERR_S	32	por_1	0x0	RW1C	Bypass access when BE = 0 secondary error
RESERVED	31:16				Reserved field
TBW_DPE_P	15	por_1	0x0	RW1C	Tablewalk data parity primary error
TBW_ERR_P	14	por_1	0x0	RW1C	Tablewalk primary error
TBW_UDE_P	13	por_1	0x0	RW1C	Tablewalk unexpected data return primary error
TBW_DME_P	12	por_1	0x0	RW1C	Tablewalk disabled miss primary error
SPARE3_P	11	por_1	0x0	RW1C	Spare primary error
SPARE2_P	10	por_1	0x0	RW1C	Spare primary error
TTC_CAE_P	9	por_1	0x0	RW1C	TTE cache CSR access primary error
TTC_DPE_P	8	por_1	0x0	RW1C	TTE cache data parity primary error

Table 1-149 MMU Error Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
TTE_PRT_P	7	por_1	0x0	RW1C	TTE write bit not set on write primary error
TTE_INV_P	6	por_1	0x0	RW1C	TTE valid bit not set primary error
TRN_OOR_P	5	por_1	0x0	RW1C	Translation access out-of-range primary error
TRN_ERR_P	4	por_1	0x0	RW1C	Translation access when TE = 0 primary error
SPARE1_P	3	por_1	0x0	RW1C	Spare primary error
SPARE0_P	2	por_1	0x0	RW1C	Spare primary error
BYP_OOR_P	1	por_1	0x0	RW1C	Bypass access out-of-range primary error
BYP_ERR_P	0	por_1	0x0	RW1C	Bypass access when BE = 0 primary error

1.3.4.51 MMU Error Status Set Register (0x00641020, 0x00741020 / 0x0)

The MMU Error Status Set Register allows for error bits to be set to simulate actual error occurrence.

Table 1-150 MMU Error Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:48				Reserved field
TBW_DPE_S	47	por_1	0x0	RW1S	Tablewalk data parity secondary error
TBW_ERR_S	46	por_1	0x0	RW1S	Tablewalk secondary error
TBW_UDE_S	45	por_1	0x0	RW1S	Tablewalk unexpected data return secondary error
TBW_DME_S	44	por_1	0x0	RW1S	Tablewalk disabled miss secondary error
SPARE3_S	43	por_1	0x0	RW1S	Spare secondary error
SPARE2_S	42	por_1	0x0	RW1S	Spare secondary error

Table 1-150 MMU Error Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
TTC_CAE_S	41	por_1	0x0	RW1S	TTE cache CSR access secondary error
TTC_DPE_S	40	por_1	0x0	RW1S	TTE cache data parity secondary error
TTE_PRT_S	39	por_1	0x0	RW1S	TTE write bit not set on write secondary error
TTE_INV_S	38	por_1	0x0	RW1S	TTE valid bit not set secondary error
TRN_OOR_S	37	por_1	0x0	RW1S	Translation access out-of-range secondary error
TRN_ERR_S	36	por_1	0x0	RW1S	Translation access when TE = 0 secondary error
SPARE1_S	35	por_1	0x0	RW1S	Spare secondary error
SPARE0_S	34	por_1	0x0	RW1S	Spare secondary error
BYP_OOR_S	33	por_1	0x0	RW1S	Bypass access out-of-range secondary error
BYP_ERR_S	32	por_1	0x0	RW1S	Bypass access when BE = 0 secondary error
RESERVED	31:16				Reserved field
TBW_DPE_P	15	por_1	0x0	RW1S	Tablewalk data parity primary error
TBW_ERR_P	14	por_1	0x0	RW1S	Tablewalk primary error
TBW_UDE_P	13	por_1	0x0	RW1S	Tablewalk unexpected data return primary error
TBW_DME_P	12	por_1	0x0	RW1S	Tablewalk disabled miss primary error
SPARE3_P	11	por_1	0x0	RW1S	Spare primary error
SPARE2_P	10	por_1	0x0	RW1S	Spare primary error

Table 1-150 MMU Error Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
TTC_CAE_P	9	por_1	0x0	RW1S	TTE cache CSR access primary error
TTC_DPE_P	8	por_1	0x0	RW1S	TTE cache data parity primary error
TTE_PRT_P	7	por_1	0x0	RW1S	TTE write bit not set on write primary error
TTE_INV_P	6	por_1	0x0	RW1S	TTE valid bit not set primary error
TRN_OOR_P	5	por_1	0x0	RW1S	Translation access out-of-range primary error
TRN_ERR_P	4	por_1	0x0	RW1S	Translation access when TE = 0 primary error
SPARE1_P	3	por_1	0x0	RW1S	Spare primary error
SPARE0_P	2	por_1	0x0	RW1S	Spare primary error
BYP_OOR_P	1	por_1	0x0	RW1S	Bypass access out-of-range primary error
BYP_ERR_P	0	por_1	0x0	RW1S	Bypass access when BE = 0 primary error

1.3.4.52 MMU Translation Fault Address Register (0x00641028, 0x00741028 / 0x0)

The MMU Translation Fault Address Register is provided to aid debug of software errors. It logs the virtual address on any primary error. This register is undefined unless a primary error is set. Diagnostic software can write this register after a primary error is set.

Table 1-151 MMU Translation Fault Address Register

Field	Bits	Reset Name	Reset Value	Type	Description
VA	63:2		62'bx	RW	Virtual address
RESERVED	1:0				Reserved field

1.3.4.53 MMU Translation Fault Status Register (0x00641030, 0x00741030 / 0x0)

The MMU Translation Fault Status Register provides the TTE cache entry address, transaction type, and Requester ID on any primary error. This register is undefined unless a primary error is set. Diagnostic software can write this register after a primary error is set.

Table 1-152 MMU Translation Fault Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:41				Reserved field
ENTRY	40:32		9'bx	RW	TTE Cache entry address
RESERVED	31:23				Reserved field
TYPE	22:16		7'bx	RW	Transaction type
ID	15:0		16'bx	RW	Requester ID

1.3.4.54 MMU Performance Counter Select Register (0x00642000, 0x00742000 / 0x0)

The MMU Performance Counter Select Register provides the selects for the performance counters.

Table 1-153 MMU Performance Counter Select Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:16				Reserved field
SEL1	15:8	rst_1	0x0	RW	Select for counter 1: 00 = None 01 = Clock cycles 02 = Total translations 03 = Total stall cycles 04 = Total translation misses 05 = Tablewalk stall cycles 06 = Bypass mode translations 07 = Translation mode translations 08 = Flow control stall cycles 09 = Total cache entries flushed
SEL0	7:0	rst_1	0x0	RW	Select for counter 0: Values are as above.

1.3.4.55 MMU Performance Counter Zero Register (0x00642008, 0x00742008 / 0x0)

The MMU Performance Counter Zero Register counts the occurrences of the events selected by the SEL0 field in the MMU Performance Counter Select Register.

Table 1-154 MMU Performance Counter Zero Register

Field	Bits	Reset Name	Reset Value	Type	Description
CNT	63:0	rst_l	0x0	RW	Counter

1.3.4.56 MMU Performance Counter One Register (0x00642010, 0x00742010 / 0x0)

The MMU Performance Counter One Register counts the occurrences of the events selected by the SEL1 field in the MMU Performance Counter Select Register

Table 1-155 MMU Performance Counter One Register

Field	Bits	Reset Name	Reset Value	Type	Description
CNT	63:0	rst_l	0x0	RW	Counter

1.3.4.57 MMU TTE Cache Virtual Tag Registers (0x00646000-0x006461F8, 0x00746000-0x007461F8 / 0x0)

The MMU TTE Cache Virtual Tag Registers give software access to the virtual tags of the TTE cache. When loading these registers for test purposes, the CNT fields should all be unique with the highest number being the valid entry which will be replaced first. Access can only occur when CM = 0 in the MMU Control and Status Register.

Access Methods: jtag_rd = HW, jtag_wr = HW, pio_slow_rd = HW, pio_slow_wr = HW, pio_med_rd = HW, pio_med_wr = HW, pio_fast_rd = HW, pio_fast_wr = HW

Table 1-156 MMU TTE Cache Virtual Tag Registers

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:44				Reserved field
CNT	43:32	rst_l	12'b0	RW	LRU count
TAG	31:16	rst_l	16'bx	RW	Virtual tag
RESERVED	15:1				Reserved field
VLD	0	rst_l	1'b0	RW	Valid

1.3.4.58 MMU TTE Cache Physical Tag Registers (0x00647000-0x006471F8, 0x00747000-0x007471F8 / 0x0)

The MMU TTE Cache Physical Tag Registers give software access to the physical tags of the TTE cache. Access can only occur when CM = 0 in the MMU Control and Status Register.

Access Methods: jtag_rd = HW, jtag_wr = HW, pio_slow_rd = HW, pio_slow_wr = HW, pio_med_rd = HW, pio_med_wr = HW, pio_fast_rd = HW, pio_fast_wr = HW

Table 1-157 MMU TTE Cache Physical Tag Registers

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:43				Reserved field
TAG	42:6	rst_l	37'bx	RW	Physical tag
RESERVED	5:1				Reserved field
VLD	0	rst_l	1'b0	RW	Valid

1.3.4.59 MMU TTE Cache Data Registers (0x00648000-0x00648FF8, 0x00748000-0x00748FF8 / 0x0)

The MMU TTE Cache Data Registers give software access to the data of the TTE cache. Access can only occur when CM = 0 in the MMU Control and Status Register. The data includes odd parity which must be calculated and written when these registers are written.

Access Methods: jtag_rd = HW, jtag_wr = HW, pio_slow_rd = HW, pio_slow_wr = HW, pio_med_rd = HW, pio_med_wr = HW, pio_fast_rd = HW, pio_fast_wr = HW

Table 1-158 MMU TTE Cache Data Registers

Field	Bits	Reset Name	Reset Value	Type	Description
PAR	63:60		4'bx	RW	Odd parity: [63] = parity for bits [42:35] [62] = parity for bits [34:27] [61] = parity for bits [26:19] [60] = parity for bits [18:13,1,0]
RESERVED	59:43				Reserved field
PPN	42:13		30'bx	RW	Physical Page Number [42:13]
RESERVED	12:2				Reserved field
WRT	1		1'bx	RW	Write
VLD	0		1'bx	RW	Valid

1.3.4.60 ILU Error Log Enable Register (0x00651000, 0x00751000 / 0xF0)

The ILU Error Log Enable Register is used to enable logging for all of the possible ILU errors detected by Fire. By default all of the errors are enabled to be logged.

Table 1-159 ILU Error Log Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:8				Reserved field
SPARE3	7	por_1	0x1	RW	Spare error
SPARE2	6	por_1	0x1	RW	Spare error
SPARE1	5	por_1	0x1	RW	Spare error
IHB_PE	4	por_1	0x1	RW	Ingress header buffer parity error
RESERVED	3:0				Reserved field

1.3.4.61 ILU Interrupt Enable Register (0x00651008, 0x00751008 / 0x0)

The ILU Interrupt Enable Register is used to enable interrupts for all of the possible ILU errors detected by Fire. By default all of the errors are disabled from generating interrupts.

Table 1-160 ILU Interrupt Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:40				Reserved field
SPARE3_S	39	rst_1	0x0	RW	Spare secondary error
SPARE2_S	38	rst_1	0x0	RW	Spare secondary error
SPARE1_S	37	rst_1	0x0	RW	Spare secondary error
IHB_PE_S	36	rst_1	0x0	RW	Ingress header buffer parity secondary error
RESERVED	35:8				Reserved field
SPARE3_P	7	rst_1	0x0	RW	Spare primary error
SPARE2_P	6	rst_1	0x0	RW	Spare primary error
SPARE1_P	5	rst_1	0x0	RW	Spare primary error

Table 1-160 ILU Interrupt Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
IHB_PE_P	4	rst_1	0x0	RW	Ingress header buffer parity primary error
RESERVED	3:0				Reserved field

1.3.4.62 ILU Interrupt Status Register (0x00651010, 0x00751010 / 0x0)

The ILU Interrupt Status Register is a read-only register software reads to obtain the source of an interrupt for the ILU block. This register will contain a 1 in all bits where an associated interrupt was generated.

Table 1-161 ILU Interrupt Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:40				Reserved field
SPARE3_S	39	rst_1	0x0	R	Spare secondary error
SPARE2_S	38	rst_1	0x0	R	Spare secondary error
SPARE1_S	37	rst_1	0x0	R	Spare secondary error
IHB_PE_S	36	rst_1	0x0	R	Ingress header buffer parity secondary error
RESERVED	35:8				Reserved field
SPARE3_P	7	rst_1	0x0	R	Spare primary error
SPARE2_P	6	rst_1	0x0	R	Spare primary error
SPARE1_P	5	rst_1	0x0	R	Spare primary error
IHB_PE_P	4	rst_1	0x0	R	Ingress header buffer parity primary error
RESERVED	3:0				Reserved field

1.3.4.63 ILU Error Status Clear Register (0x00651018, 0x00751018 / 0x0)

The ILU Error Status Clear Register is a read, write-one-to-clear register serving two purposes for software. First, software can read this register to determine which ILU errors have been logged by Fire but have not necessarily caused an interrupt (Mondo

63). This register will contain a 1 in all bits where errors were logged. Second, software uses this register to clear the ILU errors detected by Fire. It writes a 1 to the bit(s) which it wants to clear.

Table 1-162 ILU Error Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:40				Reserved field
SPARE3_S	39	por_1	0x0	RW1C	Spare secondary error
SPARE2_S	38	por_1	0x0	RW1C	Spare secondary error
SPARE1_S	37	por_1	0x0	RW1C	Spare secondary error
IHB_PE_S	36	por_1	0x0	RW1C	Ingress header buffer parity secondary error
RESERVED	35:8				Reserved field
SPARE3_P	7	por_1	0x0	RW1C	Spare primary error
SPARE2_P	6	por_1	0x0	RW1C	Spare primary error
SPARE1_P	5	por_1	0x0	RW1C	Spare primary error
IHB_PE_P	4	por_1	0x0	RW1C	Ingress header buffer parity primary error
RESERVED	3:0				Reserved field

1.3.4.64 ILU Error Status Set Register (0x00651020, 0x00751020 / 0x0)

The ILU Error Status Set Register is a read/write-one-to-set register serving two purposes for software. First, software can read this register to determine which ILU errors have been logged by Fire but have not necessarily caused an interrupt (Mondo 63). This register will contain a 1 in all bits where errors were logged. Second, software uses this register to set the ILU errors detected by Fire. It writes a 1 to the bit(s) which it wants to clear. THIS SHOULD ONLY BE USED FOR DIAG PURPOSES AND ERROR TESTING.

Table 1-163 ILU Error Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:40				Reserved field
SPARE3_S	39	por_1	0x0	RW1S	Spare secondary error
SPARE2_S	38	por_1	0x0	RW1S	Spare secondary error

Table 1-163 ILU Error Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
SPARE1_S	37	por_1	0x0	RW1S	Spare secondary error
IHB_PE_S	36	por_1	0x0	RW1S	Ingress header buffer parity secondary error
RESERVED	35:8				Reserved field
SPARE3_P	7	por_1	0x0	RW1S	Spare primary error
SPARE2_P	6	por_1	0x0	RW1S	Spare primary error
SPARE1_P	5	por_1	0x0	RW1S	Spare primary error
IHB_PE_P	4	por_1	0x0	RW1S	Ingress header buffer parity primary error
RESERVED	3:0				Reserved field

1.3.4.65 PEC Core and Block Interrupt Enable Register (0x00651800, 0x00751800 / 0x0)

This is the PEC core and block interrupt enable register.

Table 1-164 PEC Core and Block Interrupt Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
PEC	63	rst_1	0x0	RW	The mask bit to mask the pec interrupt
RESERVED	62:4				Reserved field
PEC_ILU	3	rst_1	0x0	RW	ILU block interrupt enable bit
PEC_UE	2	rst_1	0x0	RW	Uncorrectable error interrupt enable bit
PEC_CE	1	rst_1	0x0	RW	Correctable error interrupt enable bit
PEC_OE	0	rst_1	0x0	RW	Other event interrupt enable bit

1.3.4.66 PEC Core and Block Interrupt Status Register (0x00651808, 0x00751808 / 0x0)

This is the PEC core and block interrupt status register.

Table 1-165 PEC Core and Block Interrupt Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:4				Reserved field
ILU	3	rst_1	0x0	R	ILU header parity error
UE	2	rst_1	0x0	R	Uncorrectable error from PEC
CE	1	rst_1	0x0	R	Correctable error from PEC
OE	0	rst_1	0x0	R	Other event from PEC

1.3.4.67 ILU Device Capabilities Register (0x00652000, 0x00752000 / 0x0)

The reset value is the default for systems not supporting EStar. SW must set this register for the systems supporting EStar.

Table 1-166 ILU Device Capabilities Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:1				Reserved field
ESTAR	0	rst_1	0x0	RW	System EStar Support 0b = System EStar not supported 1b = System EStar supported

1.3.4.68 DMC Debug Select Register for Port A (0x00653000, 0x00753000 / 0x0)

Table 1-167 DMC Debug Select Register for Port A

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:10				Reserved field

Table 1-167 DMC Debug Select Register for Port A

Field	Bits	Reset Name	Reset Value	Type	Description
BLOCK_SEL	9:6	rst_l	0x0	RW	DMC Block Debug Selects for Port A 0000 - All Zeros 0001 - CLU Block Selects 0010 - CMU Block Selects 0011 - CRU Block Selects 0100 - All Zeros 0101 - All Zeros 0110 - ILU Block Selects 0111 - All Zeros 1000 - All Zeros 1001 - IMU Block Selects 1010 - MMU Block Selects 1011 - PMU Block Selects 1100 - PSB Block Selects 1101 - RMU Block Selects 1110 - TMU Block Selects 1111 - TSB Block Selects
SUB_SEL	5:3	rst_l	0x0	RW	Select the sub-block for Port A
SIGNAL_SEL	2:0	rst_l	0x0	RW	Select the signals for Port A

1.3.4.69 DMC Debug Select Register for Port B (0x00653008, 0x00753008 / 0x0)

Table 1-168 DMC Debug Select Register for Port B

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:10				Reserved field

Table 1-168 DMC Debug Select Register for Port B

Field	Bits	Reset Name	Reset Value	Type	Description
BLOCK_SEL	9:6	rst_l	0x0	RW	DMC Block Debug Selects for Port B 0000 - All Zeros 0001 - CLU Block Selects 0010 - CMU Block Selects 0011 - CRU Block Selects 0100 - All Zeros 0101 - All Zeros 0110 - ILU Block Selects 0111 - All Zeros 1000 - All Zeros 1001 - IMU Block Selects 1010 - MMU Block Selects 1011 - PMU Block Selects 1100 - PSB Block Selects 1101 - RMU Block Selects 1110 - TMU Block Selects 1111 - TSB Block Selects
SUB_SEL	5:3	rst_l	0x0	RW	Select sub-block for Port B
SIGNAL_SEL	2:0	rst_l	0x0	RW	Select the signals for Port B

1.3.4.70 DMC PCI Express Configuration Register (0x00653100, 0x00753100 / 0x0)

The DMC PCI Express Configuration Register specifies the Requester ID used for FIRE and the Bus Number used for Type 0 Configuration Requests.

Table 1-169 DMC PCI Express Configuration Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
BUS_NUM	31:24	rst_l	0x0	RW	Bus Number used for Type 0 Configuration Requests
RESERVED	23:16				Reserved field

Table 1-169 DMC PCI Express Configuration Register

Field	Bits	Reset Name	Reset Value	Type	Description
REQ_ID	15:0	rst_l	0x0	RW	Requester ID used by FIRE. NOTE: do not change this default value, otherwise, Fire's outgoing Message Requests would not have a proper RequesterID value

1.3.4.71 Packet Scoreboard DMA Register Set (0x00660000-0x006600F8, 0x00760000-0x007600F8 / 0x0)

The Packet Scoreboard DMA Register Set comprises 32 register sets of 41 bits each for storing and retrieving DMA record information associated with a given packet

Table 1-170 Packet Scoreboard DMA Register Set

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:41				Reserved field
ENTRY	40:0	rst_l	0x0	R	[40:36] Transaction Tag [35:31] Context Number [30:26] Packet Seq. Number [25:22] Cacheline Total [21:12] Length [11:0] Byte Count

1.3.4.72 Packet Scoreboard PIO Register Set (0x00664000-0x00664078, 0x00764000-0x00764078 / 0x0)

The Packet Scoreboard PIO Register Set comprises 16 register sets of 6 bits each for storing and retrieving PIO Record Information associated with a given packet

Table 1-171 Packet Scoreboard PIO Register Set

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:6				Reserved field
ENTRY	5:0	rst_l	0x0	R	[5:2] Transaction ID [1:0] Agent ID

1.3.4.73 Transaction Scoreboard Register Set (0x00670000-0x006700F8, 0x00770000-0x007700F8 / 0x0)

The Transaction Scoreboard Register Set comprises 32 register sets of 48 bits each for storing and retrieving DMA read record information associated with a given transaction

Table 1-172 Transaction Scoreboard Register Set

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:48				Reserved field
ENTRY	47:0	rst_1	0x0	R	[47:45] Traffic Class [44:43] Attribute [42:31] Byte Count [30:15] Requester ID [14:7] Transaction Layer Packet [6:0] Address Alignment Bits

1.3.4.74 Transaction Scoreboard Status Register (0x00670100, 0x00770100 / 0x1)

The Transaction Scoreboard Status Register reports status on pending DMAs.

Table 1-173 Transaction Scoreboard Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:8				Reserved field
FULL	7	rst_1	0x0	R	TSB fullness 1'b1 - TSB is full 1'b0 - TSB is not full
NUM_PND_DMA	6:1	rst_1	0x0	R	Number of pending DMA transactions
EMPTY	0	rst_1	0x1	R	TSB emptiness 1'b1 - TSB is empty (no pending DMAs) 1'b0 - TSB is not empty

1.3.4.75 TLU Control Register (0x00680000, 0x00780000 / 0x101)

The TLU Control Register provides controls for the transaction layer. It also provides configuration for the link layer

Table 1-174 TLU Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
LOS_TIM	31:24	por_1	0x0	RW	L0s Entry Timer Value. Values are in multiples of 32 ns. i.e. 00h = 0 ns 01h = 32 ns 02h = 64 ns DAh = 7.0 us (recommended) FFh = 8.2 us
RESERVED	23:21				Reserved field
NPWR_EN	20	por_1	0x0	RW	Non-posted Write Enable 0b - Issue non-posted writes at will 1b - Non-posted write will stall issue of all other requests until it completes
RESERVED	19				Reserved field
CTO_SEL	18:16	por_1	0x0	RW	Completion Timeout Select 000b - Infinite 001b - 2 ²⁶ symbol times (268 ms) 010b - 2 ²⁵ symbol times (134 ms) 011b - 2 ²⁴ symbol times (67.1 ms) 100b - 2 ²³ symbol times (33.6 ms) 101b - 2 ²² symbol times (16.8 ms) 110b - 2 ²¹ symbol times (8.4 ms) 111b - 2 ⁹ symbol times (2 us)

Table 1-174 TLU Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
CONFIG	15:0	por_1	0x101	RW	<p>Configuration</p> <p>[0] - Port</p> <p>0b - Upstream Port</p> <p>1b - Downstream Port</p> <p>[1] - Common Clock Configuration</p> <p>0b - This component and the component at the opposite end of this link are operating with asynchronous reference clock.</p> <p>1b - This component and the component at the opposite end of this Link are operating with a distributed common reference clock.</p> <p>[4:2] - Autoconfigure Max Payload Size</p> <p>This is the MPS used by the link to auto configure timers. Software should manually configure timers and not change this field from the default. 000b - 128 B</p> <p>001b - 256 B</p> <p>010b - 512 B</p> <p>011b - 1024 B</p> <p>100b - 2048 B</p> <p>101b - 4096 B</p> <p>110b - Reserved</p> <p>111b - Reserved</p> <p>[5] - Digital Loopback Mode</p> <p>0b - Digital loopback mode disabled.</p> <p>1b - Digital loopback mode enabled.</p> <p>[6] - Ewrap Loopback Mode</p> <p>0b - Ewrap loopback mode disabled.</p> <p>1b - Ewrap loopback mode enabled.</p> <p>[7] - Pad Loopback Mode</p> <p>0b - Pad loopback mode disabled.</p> <p>1b - Pad loopback mode enabled.</p> <p>[8] - Remain in Detect.Quiet</p> <p>0b - Proceed with link training.</p> <p>1b - Remain in state Detect.Quiet.</p> <p>[15:9] - Reserved</p>

1.3.4.76 TLU Status Register (0x00680008, 0x00780008 / 0x0)

The TLU Status Register reports the status from the link layer

Table 1-175 TLU Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:9				Reserved field
DRAIN	8	rst_1	0x0	RW1C	Drain state Set by hardware when in the drain state. Cleared by software when preparing to bring the link back up.
STATUS	7:0	rst_1	0x0	R	Status [2:0] - Data Link State 001b - Data Link Inactive 010b - Data Link Init 100b - Data Link Active [3] - Recovery [7:4] - Reserved

1.3.4.77 TLU PME Turn Off Generate Register (0x00680010, 0x00780010 / 0x0)

The TLU PME Turn Off Generate Register generates a PME_turn_off message

Table 1-176 TLU PME Turn Off Generate Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:1				Reserved field
PTO	0	rst_1	0x0	RW1S	Generate PME_turn_off message 0b - Not Pending 1b - Pending

1.3.4.78 TLU Ingress Credits Initial Register (0x00680018, 0x00780018 / 0x10000200C0)

The TLU Ingress Credits Initial Register specifies the initial credits which are advertised. The Completion Header Credit, Completion Data Credit, and Non-posted Data Credit are all advertised as infinite and cannot be changed. The sum of the Non-

posted Header Credit and Posted Header Credit must be less than or equal to 0x30. The Posted Data Credit must be less than or equal to 0x0C0. The values only take effect when the link is down.

Table 1-177 TLU Ingress Credits Initial Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:60				Reserved field
CHC	59:52	por_1	0x0	R	Completion Header Credit
CDC	51:40	por_1	0x0	R	Completion Data Credit
NHC	39:32	por_1	0x10	RW	Non-posted Header Credit
NDC	31:20	por_1	0x0	R	Non-posted Data Credit
PHC	19:12	por_1	0x20	RW	Posted Header Credit
PDC	11:0	por_1	0xC0	RW	Posted Data Credit

1.3.4.79 TLU Diagnostic Register (0x00680100, 0x00780100 / 0x0)

The TLU Diagnostic Register allows for error injection and other diagnostic functions.

Table 1-178 TLU Diagnostic Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:54				Reserved field
LNK_MAX	53:48	rst_1	0x0	RW	Starting link width for training: 000000b - use default 000001b - x1 000100b - x4 001000b - x8 all other encodings are reserved.

Table 1-178 TLU Diagnostic Register

Field	Bits	Reset Name	Reset Value	Type	Description
CHK_DIS	47:32	rst_1	0x0	RW	Disable error checks: [32] - Flow control max credit [33] - Flow control finite update [34] - Reserved [35] - Reserved [36] - Byte enable rules [37] - 4KB boundary crossing [38] - Receiver overflow [39] - Reserved [40] - Non-configuration with CRS [41] - TC mismatch in Cpl(D) [42] - Attribute mismatch in Cpl(D) [43] - Byte count mismatch in Cpl(D) [44] - Lower address mismatch in Cpl(D) [45] - Reserved [46] - Reserved [47] - Reserved
RESERVED	31:24				Reserved field
EPI_PAR	23:16	rst_1	0x0	RW	Egress Parity Invert Bits
IDI_PAR	15:12	rst_1	0x0	RW	Ingress Data Parity Invert Bits
IHI_PAR	11:8	rst_1	0x0	RW	Ingress Header Parity Invert Bits
EPI_TRG	7	rst_1	0x0	RW1S	Egress Parity Invert Trigger When set the next valid cycle on the Egress Transaction Packet interface will invert parity based upon the value of the Egress Parity Invert Bits. This bit will be cleared when this occurs

Table 1-178 TLU Diagnostic Register

Field	Bits	Reset Name	Reset Value	Type	Description
IDI_TRG	6	rst_1	0x0	RW1S	Ingress Data Parity Invert Trigger When set the next valid write into the Ingress Data Buffer will invert parity based upon the value of the Ingress Data Parity Invert Bits. This bit will be cleared when this occurs.
IHI_TRG	5	rst_1	0x0	RW1S	Ingress Header Parity Invert Trigger When set the next valid write into the Ingress Header Buffer will invert parity based upon the value of the Ingress Header Parity Invert Bits. This bit will be cleared when this occurs
MRC_TRG	4	rst_1	0x0	RW1S	Memory Read Capture Trigger When this bit is set the next Memory Read Request will be removed from the pipeline and if enabled captured in the Receive Other Event Header Log Registers. This bit will be cleared when this occurs.
RESERVED	3:2				Reserved field
EPP_DIS	1	rst_1	0x0	RW	Egress Packet Processing Disable When this bit is set, no packets will be transmitted.
IFC_DIS	0	rst_1	0x0	RW	Ingress Flow Control Update Disable When this bit is set, the Ingress Flow Control Allocated Credits will not get updated when they become available. When this bit is subsequently cleared, credits will be restored as normal.

1.3.4.80 TLU Egress Credits Consumed Register (0x00680200, 0x00780200 / 0x0)

The TLU Egress Credits Consumed Register indicates the current credits consumed by the egress block. It also indicates if any of the header credits were advertised being infinite.

Table 1-179 TLU Egress Credits Consumed Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63				Reserved field
CHI	62		1'bx	R	Infinite Completion Header Credits
NHI	61		1'bx	R	Infinite Non-posted Header Credits
PHI	60		1'bx	R	Infinite Posted Header Credits
CHC	59:52		8'bx	R	Completion Header Credit
CDC	51:40		12'bx	R	Completion Data Credit
NHC	39:32		8'bx	R	Non-posted Header Credit
NDC	31:20		12'bx	R	Non-posted Data Credit
PHC	19:12		8'bx	R	Posted Header Credit
PDC	11:0		12'bx	R	Posted Data Credit

1.3.4.81 TLU Egress Credit Limit Register (0x00680208, 0x00780208 / 0x0)

The TLU Egress Credit Limit Register indicates the current credit limit advertised to the egress block. It also indicates if any of the data credits were advertised being infinite.

Table 1-180 TLU Egress Credit Limit Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63				Reserved field
CDI	62		1'bx	R	Infinite Completion Data Credits
NDI	61		1'bx	R	Infinite Non-posted Data Credits
PDI	60		1'bx	R	Infinite Posted Data Credits
CHC	59:52		8'bx	R	Completion Header Credit

Table 1-180 TLU Egress Credit Limit Register

Field	Bits	Reset Name	Reset Value	Type	Description
CDC	51:40		12'bx	R	Completion Data Credit
NHC	39:32		8'bx	R	Non-posted Header Credit
NDC	31:20		12'bx	R	Non-posted Data Credit
PHC	19:12		8'bx	R	Posted Header Credit
PDC	11:0		12'bx	R	Posted Data Credit

1.3.4.82 TLU Egress Retry Buffer Register (0x00680210, 0x00780210 / 0x0)

The TLU Egress Retry Buffer Register indicates the credits consumed and the credit limit for the retry buffer.

Table 1-181 TLU Egress Retry Buffer Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:48				Reserved field
CC	47:32		16'bx	R	Credits Consumed
RESERVED	31:16				Reserved field
CL	15:0		16'bx	R	Credit Limit

1.3.4.83 TLU Ingress Credits Allocated Register (0x00680218, 0x00780218 / 0x0)

The TLU Ingress Credits Allocated Register indicates the credits currently allocated by the ingress block.

Table 1-182 TLU Ingress Credits Allocated Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:60				Reserved field
CHC	59:52		8'bx	R	Completion Header Credit
CDC	51:40		12'bx	R	Completion Data Credit
NHC	39:32		8'bx	R	Non-posted Header Credit
NDC	31:20		12'bx	R	Non-posted Data Credit

Table 1-182 TLU Ingress Credits Allocated Register

Field	Bits	Reset Name	Reset Value	Type	Description
PHC	19:12		8'bx	R	Posted Header Credit
PDC	11:0		12'bx	R	Posted Data Credit

1.3.4.84 TLU Ingress Credits Received Register (0x00680220, 0x00780220 / 0x0)

The LTU Ingress Credits Received Register indicated the credits currently received by the ingress block.

Table 1-183 TLU Ingress Credits Received Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:60				Reserved field
CHC	59:52		8'bx	R	Completion Header Credit
CDC	51:40		12'bx	R	Completion Data Credit
NHC	39:32		8'bx	R	Non-posted Header Credit
NDC	31:20		12'bx	R	Non-posted Data Credit
PHC	19:12		8'bx	R	Posted Header Credit
PDC	11:0		12'bx	R	Posted Data Credit

1.3.4.85 TLU Other Event Log Enable Register (0x00681000, 0x00781000 / 0xFFFFF)

The TLU Other Event Log Enable Register controls which events will be logged in the TLU Logged Other Event Status Register.

Table 1-184 TLU Other Event Log Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:24				Reserved field
EN	23:0	por_1	0xFFFF FF	RW	Log Enables

1.3.4.86 TLU Other Event Interrupt Enable Register (0x00681008, 0x00781008 / 0x0)

The TLU Other Event Interrupt Enable Register controls which events logged in the TLU Logged Other Event Status Register will generate an interrupt

Table 1-185 TLU Other Event Interrupt Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:56				Reserved field
EN_S	55:32	rst_l	0x0	RW	Secondary Interrupt Enables
RESERVED	31:24				Reserved field
EN_P	23:0	rst_l	0x0	RW	Primary Interrupt Enables

1.3.4.87 TLU Other Event Interrupt Status Register (0x00681010, 0x00781010 / 0x0)

The TLU Other Event Interrupt Status Register indicates which events logged in the TLU Logged Other Event Status Register are enabled to generate interrupts by the TLU Other Event Interrupt Enable Register

Table 1-186 TLU Other Event Interrupt Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:56				Reserved field
ERR_S	55:32	rst_l	0x0	R	Secondary Interrupt Status
RESERVED	31:24				Reserved field
ERR_P	23:0	rst_l	0x0	R	Primary Interrupt Status

1.3.4.88 TLU Other Event Status Clear Register (0x00681018, 0x00781018 / 0x0)

The TLU Other Event Status Clear Register indicates that an event occurred. Primary events include data logged with these events. Secondary events do not have any data logged with them. All events belong to the same event group which means that any primary event logged will cause any additional event to get logged as secondary.

Table 1-187 TLU Other Event Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:56				Reserved field
SPARE_S	55	por_l	0x0	RW1C	Spare Secondary Event

Table 1-187 TLU Other Event Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
MFC_S	54	por_1	0x0	RW1C	Malformed Completion Secondary Event
CTO_S	53	por_1	0x0	RW1C	Completion Timeout Secondary Event
NFP_S	52	por_1	0x0	RW1C	No Forward Progress Secondary Event Set when no TLP is successfully received between two consecutive recovery attempts.
LWC_S	51	por_1	0x0	RW1C	Potential Link Width Change Secondary Event on LTSSM re-configuration due to recovery failure
MRC_S	50	por_1	0x0	RW1C	Memory Read Capture Primary Event
WUC_S	49	por_1	0x0	RW1C	Write Unsuccessful Completion Status Secondary Event
RUC_S	48	por_1	0x0	RW1C	Read Unsuccessful Completion Status Secondary Event
CRS_S	47	por_1	0x0	RW1C	Configuration Request Retry Completion Status Secondary Event
IIP_S	46	por_1	0x0	RW1C	Ingress Interface Parity Error Secondary Event
EDP_S	45	por_1	0x0	RW1C	Egress Data Parity Error Secondary Event
EHP_S	44	por_1	0x0	RW1C	Egress Header Parity Error Secondary Event
LIN_S	43	por_1	0x0	RW1C	Link Interrupt Secondary Event This is the path taken on any LPU unmasked interrupt.
LRS_S	42	por_1	0x0	RW1C	Link Reset Secondary Event
LDN_S	41	por_1	0x0	RW1C	Link Down Secondary Event

Table 1-187 TLU Other Event Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
LUP_S	40	por_1	0x0	RW1C	Link Up Secondary Event
LPU_S	39:38	por_1	0x0	RW1C	Spare Secondary Events
ERU_S	37	por_1	0x0	RW1C	Egress Retry Buffer Underflow Error Secondary Event
ERO_S	36	por_1	0x0	RW1C	Egress Retry Buffer Overflow Error Secondary Event
EMP_S	35	por_1	0x0	RW1C	Egress Minimum Packet Error Secondary Event
EPE_S	34	por_1	0x0	RW1C	Egress protocol Error Secondary Event
ERP_S	33	por_1	0x0	RW1C	Egress Retry Parity Error Secondary Event
EIP_S	32	por_1	0x0	RW1C	Egress Interface Parity Error Secondary Event
RESERVED	31:24				Reserved field
SPARE_P	23	por_1	0x0	RW1C	Spare Primary Event
MFC_P	22	por_1	0x0	RW1C	Malformed Completion Primary Event
CTO_P	21	por_1	0x0	RW1C	Completion Timeout Primary Event
NFP_P	20	por_1	0x0	RW1C	No Forward Progress Primary Event Set when no TLP is successfully received between two consecutive recovery attempts.
LWC_P	19	por_1	0x0	RW1C	Potential Link Width Change Primary Event on LTSSM re-configuration due to recovery failure
MRC_P	18	por_1	0x0	RW1C	Memory Read Capture Primary Event

Table 1-187 TLU Other Event Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
WUC_P	17	por_1	0x0	RW1C	Write Unsuccessful Completion Status Primary Event
RUC_P	16	por_1	0x0	RW1C	Read Unsuccessful Completion Status Primary Event
CRS_P	15	por_1	0x0	RW1C	Configuration Request Retry Completion Status Primary Event
IIP_P	14	por_1	0x0	RW1C	Ingress Interface Parity Error Primary Event
EDP_P	13	por_1	0x0	RW1C	Egress Data Parity Error Primary Event
EHP_P	12	por_1	0x0	RW1C	Egress Header Parity Error Primary Event
LIN_P	11	por_1	0x0	RW1C	Link Interrupt Primary Event This is the path taken on any LPU unmasked interrupt.
LRS_P	10	por_1	0x0	RW1C	Link Reset Primary Event
LDN_P	9	por_1	0x0	RW1C	Link Down Primary Event
LUP_P	8	por_1	0x0	RW1C	Link Up Primary Event
LPU_P	7:6	por_1	0x0	RW1C	Spare Primary Events
ERU_P	5	por_1	0x0	RW1C	Egress Retry Buffer Underflow Error Primary Event
ERO_P	4	por_1	0x0	RW1C	Egress Retry Buffer Overflow Error Primary Event
EMP_P	3	por_1	0x0	RW1C	Egress Minimum Packet Error Primary Event
EPE_P	2	por_1	0x0	RW1C	Egress protocol Error Primary Event
ERP_P	1	por_1	0x0	RW1C	Egress Retry Parity Error Primary Event

Table 1-187 TLU Other Event Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
EIP_P	0	por_1	0x0	RW1C	Egress Interface Parity Error Primary Event

1.3.4.89 TLU Other Event Status Set Register (0x00681020, 0x00781020 / 0x0)

The TLU Other Event Status Set Register controls setting events in the TLU Other Event Status Clear Register for testing purposes.

Table 1-188 TLU Other Event Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:56				Reserved field
SPARE_S	55	por_1	0x0	RW1S	Spare Secondary Event
MFC_S	54	por_1	0x0	RW1S	Malformed Completion Secondary Event
CTO_S	53	por_1	0x0	RW1S	Completion Timeout Secondary Event
NFP_S	52	por_1	0x0	RW1S	No Forward Progress Secondary Event Set when no TLP is successfully received between two consecutive recovery attempts.
LWC_S	51	por_1	0x0	RW1S	Potential Link Width Change Secondary Event on LTSSM re-configuration due to recovery failure
MRC_S	50	por_1	0x0	RW1S	Memory Read Capture Primary Event
WUC_S	49	por_1	0x0	RW1S	Write Unsuccessful Completion Status Secondary Event
RUC_S	48	por_1	0x0	RW1S	Read Unsuccessful Completion Status Secondary Event
CRS_S	47	por_1	0x0	RW1S	Configuration Request Retry Completion Status Secondary Event

Table 1-188 TLU Other Event Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
IIP_S	46	por_1	0x0	RW1S	Ingress Interface Parity Error Secondary Event
EDP_S	45	por_1	0x0	RW1S	Egress Data Parity Error Secondary Event
EHP_S	44	por_1	0x0	RW1S	Egress Header Parity Error Secondary Event
LIN_S	43	por_1	0x0	RW1S	Link Interrupt Secondary Event This is the path taken on any LPU unmasked interrupt.
LRS_S	42	por_1	0x0	RW1S	Link Reset Secondary Event
LDN_S	41	por_1	0x0	RW1S	Link Down Secondary Event
LUP_S	40	por_1	0x0	RW1S	Link Up Secondary Event
LPU_S	39:38	por_1	0x0	RW1S	Spare Secondary Events
ERU_S	37	por_1	0x0	RW1S	Egress Retry Buffer Underflow Error Secondary Event
ERO_S	36	por_1	0x0	RW1S	Egress Retry Buffer Overflow Error Secondary Event
EMP_S	35	por_1	0x0	RW1S	Egress Minimum Packet Error Secondary Event
EPE_S	34	por_1	0x0	RW1S	Egress protocol Error Secondary Event
ERP_S	33	por_1	0x0	RW1S	Egress Retry Parity Error Secondary Event
EIP_S	32	por_1	0x0	RW1S	Egress Interface Parity Error Secondary Event
RESERVED	31:24				Reserved field
SPARE_P	23	por_1	0x0	RW1S	Spare Primary Event
MFC_P	22	por_1	0x0	RW1S	Malformed Completion Primary Event

Table 1-188 TLU Other Event Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
CTO_P	21	por_1	0x0	RW1S	Completion Timeout Primary Event
NFP_P	20	por_1	0x0	RW1S	No Forward Progress Primary Event Set when no TLP is successfully received between two consecutive recovery attempts.
LWC_P	19	por_1	0x0	RW1S	Potential Link Width Change Primary Event on LTSSM re-configuration due to recovery failure
MRC_P	18	por_1	0x0	RW1S	Memory Read Capture Primary Event
WUC_P	17	por_1	0x0	RW1S	Write Unsuccessful Completion Status Primary Event
RUC_P	16	por_1	0x0	RW1S	Read Unsuccessful Completion Status Primary Event
CRS_P	15	por_1	0x0	RW1S	Configuration Request Retry Completion Status Primary Event
IIP_P	14	por_1	0x0	RW1S	Ingress Interface Parity Error Primary Event
EDP_P	13	por_1	0x0	RW1S	Egress Data Parity Error Primary Event
EHP_P	12	por_1	0x0	RW1S	Egress Header Parity Error Primary Event
LIN_P	11	por_1	0x0	RW1S	Link Interrupt Primary Event This is the path taken on any LPU unmasked interrupt.
LRS_P	10	por_1	0x0	RW1S	Link Reset Primary Event
LDN_P	9	por_1	0x0	RW1S	Link Down Primary Event
LUP_P	8	por_1	0x0	RW1S	Link Up Primary Event

Table 1-188 TLU Other Event Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
LPU_P	7:6	por_1	0x0	RW1S	Spare Primary Events
ERU_P	5	por_1	0x0	RW1S	Egress Retry Buffer Underflow Error Primary Event
ERO_P	4	por_1	0x0	RW1S	Egress Retry Buffer Overflow Error Primary Event
EMP_P	3	por_1	0x0	RW1S	Egress Minimum Packet Error Primary Event
EPE_P	2	por_1	0x0	RW1S	Egress protocol Error Primary Event
ERP_P	1	por_1	0x0	RW1S	Egress Retry Parity Error Primary Event
EIP_P	0	por_1	0x0	RW1S	Egress Interface Parity Error Primary Event

1.3.4.90 TLU Receive Other Event Header1 Log Register (0x00681028, 0x00781028 / 0x0)

The TLU Receive Other Event Header1 Log Register holds the first eight bytes of the received header when one of the following primary other events in the TLU Logged Other Event Error Status Register is set:

Memory Read Capture

Write Unsuccessful Completion Status

Read Unsuccessful Completion Status

Configuration Request Retry Completion Status

Table 1-189 TLU Receive Other Event Header1 Log Register

Field	Bits	Reset Name	Reset Value	Type	Description
HDR	63:0		64'bx	RW	[63:56] - Header Byte 0 [55:48] - Header Byte 1 [47:40] - Header Byte 2 [39:32] - Header Byte 3 [31:24] - Header Byte 4 [23:16] - Header Byte 5 [15:8] - Header Byte 6 [7:0] - Header Byte 7

1.3.4.91 TLU Receive Other Event Header2 Log Register (0x00681030, 0x00781030 / 0x0)

The TLU Receive Other Event Header1 Log Register holds the second eight bytes of the received header when one of the following primary other events in the TLU Logged Other Event Error Status Register is set:

Memory Read Capture

Write Unsuccessful Completion Status

Read Unsuccessful Completion Status

Configuration Request Retry Completion Status

Table 1-190 TLU Receive Other Event Header2 Log Register

Field	Bits	Reset Name	Reset Value	Type	Description
HDR	63:0		64'bx	RW	[63:56] - Header Byte 8 [55:48] - Header Byte 9 [47:40] - Header Byte A [39:32] - Header Byte B [31:24] - Header Byte C or Undefined [23:16] - Header Byte D or Undefined [15:8] - Header Byte E or Undefined [7:0] - Header Byte F or Undefined

1.3.4.92 TLU Transmit Other Event Header1 Log Register (0x00681038, 0x00781038 / 0x0)

The TLU Transmit Other Event Header1 Log Register holds the first eight bytes of the original transmitted request header when one of the following primary other events in the TLU Logged Other Event Error Status Register is set on a completion:

Write Unsuccessful Completion Status

Read Unsuccessful Completion Status

Configuration Request Retry Completion Status

Table 1-191 TLU Transmit Other Event Header1 Log Register

Field	Bits	Reset Name	Reset Value	Type	Description
HDR	63:0		64'bx	RW	[63:56] - Header Byte 0 [55:48] - Header Byte 1 [47:40] - Header Byte 2 [39:32] - Header Byte 3 [31:24] - Header Byte 4 [23:16] - Header Byte 5 [15:8] - Header Byte 6 [7:0] - Header Byte 7

1.3.4.93 TLU Transmit Other Event Header2 Log Register (0x00681040, 0x00781040 / 0x0)

The TLU Transmit Other Event Header1 Log Register holds the second eight bytes of the original transmitted request header when one of the following primary other events in the TLU Logged Other Event Error Status Register is set on a completion:

Write Unsuccessful Completion Status

Read Unsuccessful Completion Status

Configuration Request Retry Completion Status

Table 1-192 TLU Transmit Other Event Header2 Log Register

Field	Bits	Reset Name	Reset Value	Type	Description
HDR	63:0		64'bx	RW	[63:56] - Header Byte 8 [55:48] - Header Byte 9 [47:40] - Header Byte A [39:32] - Header Byte B [31:24] - Header Byte C or Undefined [23:16] - Header Byte D or Undefined [15:8] - Header Byte E or Undefined [7:0] - Header Byte F or Undefined

1.3.4.94 TLU Performance Counter Select Register (0x00682000, 0x00782000 / 0x0)

The TLU Performance Counter Select Register provides the selects for the performance counters.

Table 1-193 TLU Performance Counter Select Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:18				Reserved field
SEL2	17:16	rst_1	0x0	RW	<p>Select for counter 2:</p> <p>00b = None</p> <p>01h = Total non-posted completion time in unit of 64-cycle.</p> <p>used together with 02h (comple- tions received) in sel1 to figure out the average PIO completion latency.</p> <p>The deviation of PIO latency ranges from 0 to 128 cycles.</p> <p>02h = Transmitted data words</p> <p>03h = Received data words</p>

Table 1-193 TLU Performance Counter Select Register

Field	Bits	Reset Name	Reset Value	Type	Description
SEL1	15:8	rst_1	0x0	RW	Select for counter 1: 00h = None (counter retains previous value) 01h = Clock cycles 02h = Completions received 10h = Transmit posted credit not available cycles 11h = Transmit non-posted credit not available cycles 12h = Transmit completion credit not available cycles 13h = Transmit credits (any) not available cycles 14h = Retry buffer credit not available cycles 20h = Received memory read packets 21h = Received memory write packets 22h = Receive credits below threshold cycles 23h = Receive posted header credits exhausted cycles 24h = Receive posted data credits below MPS threshold cycles 25h = Receive non-posted header credits exhausted cycles 30h = L0s cycles (receiver) 31h = L0s transitions (receiver) 32h = L0s cycles (transmitter) 33h = L0s transitions (transmitter) 40h = Receiver Errors 42h = Bad TLPs 43h = Bad DLLPs 44h = REPLAY_NUM Rollovers 47h = Replay Timeouts
SEL0	7:0	rst_1	0x0	RW	Select for counter 0: Values are the same as counter1.

1.3.4.95 TLU Performance Counter Zero Register (0x00682008, 0x00782008 / 0x0)

The TLU Performance Counter Zero Register contains the current count of events selected by the sel0 field in the TLU Performance Counter Control Register.

Table 1-194 TLU Performance Counter Zero Register

Field	Bits	Reset Name	Reset Value	Type	Description
CNT	63:0	rst_l	0x0	RW	Counter

1.3.4.96 TLU Performance Counter One Register (0x00682010, 0x00782010 / 0x0)

The TLU Performance Counter One Register contains the current count of events selected by the sel1 field in the TLU Performance Counter Control Register.

Table 1-195 TLU Performance Counter One Register

Field	Bits	Reset Name	Reset Value	Type	Description
CNT	63:0	rst_l	0x0	RW	Counter

1.3.4.97 TLU Performance Counter Two Register (0x00682018, 0x00782018 / 0x0)

The TLU Performance Counter Two Register contains the current count of data selected by the sel2 field in the TLU Performance Counter Control Register.

Table 1-196 TLU Performance Counter Two Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
CNT	31:0	rst_l	0x0	RW	Counter

1.3.4.98 TLU Debug Select A Register (0x00683000, 0x00783000 / 0x0)

The Debug Select A Register selects the output on debug port 0

Table 1-197 TLU Debug Select A Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:9				Reserved field

Table 1-197 TLU Debug Select A Register

Field	Bits	Reset Name	Reset Value	Type	Description
BLOCK	8:6	rst_l	0x0	RW	Block select in core 000b - Constant zero 001b - Test modes 010b - ETL block 011b - ITL block 100b - PMC block 101b - RSB block 110b - CTB block 111b - LPU core
MODULE	5:3	rst_l	0x0	RW	Module select in block
SIGNAL	2:0	rst_l	0x0	RW	Signal select in sub-block

1.3.4.99 TLU Debug Select B Register (0x00683008, 0x00783008 / 0x0)

The Debug Select B Register selects the output on debug port 1

Table 1-198 TLU Debug Select B Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:9				Reserved field
BLOCK	8:6	rst_l	0x0	RW	Block select in core 000 - Constant zero 001 - Test modes 010 - ETL block 011 - ITL block 100 - PMC block 101 - RSM block 110 - CTB block 111 - LPU core
MODULE	5:3	rst_l	0x0	RW	Module select in block
SIGNAL	2:0	rst_l	0x0	RW	Signal select in module

1.3.4.100 TLU Device Capabilities Register (0x00690000, 0x00790000 / 0x2)

The TLU Device Capabilities Register identifies PCI Express device specific capabilities.

Table 1-199 TLU Device Capabilities Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:12				Reserved field
L1	11:9	rst_1	0x0	R	Endpoint L1 Acceptable Latency - This field indicated the acceptable latency that can be withstood due to the transition from L1 state to L0 state. It is essentially an indirect measure of the internal buffering. This field must be 000b for root complex.
L0S	8:6	rst_1	0x0	R	Endpoint L0s Acceptable Latency - This field indicates the acceptable latency that can be withstood due to the transition from L0s state to L0 state. It is essentially an indirect measure of the internal buffering. This field must be 000b for root complex.
RESERVED	5:3				Reserved field
MPS	2:0	rst_1	0x2	R	Max_Payload_Size Supported - This field indicates the maximum payload size that can be supported for TLPs. 010b = 512 bytes

1.3.4.101 TLU Device Control Register (0x00690008, 0x00790008 / 0x0)

The Device Control Register controls PCI Express device specific parameters.

Table 1-200 TLU Device Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:15				Reserved field

Table 1-200 TLU Device Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
MRRS	14:12	rst_l	0x0	R	Max_Read_Request_Size - This field sets the maximum Read Request size for the device as a Requester. Since the maximum Read Request size which can be issued is 64 bytes the value is 000b.
RESERVED	11:8				Reserved field
MPS	7:5	rst_l	0x0	RW	Max_Payload_Size - This field sets maximum TLP payload size for the device. As a receiver, the device must handle TLPs as large as the set value; as transmitter, the device must not generate TLPs exceeding the set value. 000b = 128 bytes 001b = 256 bytes 010b = 512 bytes 011b = 1024 bytes (not supported) 100b = 2048 bytes (not supported) 101b = 4096 bytes (not supported) 110b = reserved 111b = reserved
RESERVED	4:0				Reserved field

1.3.4.102 TLU Device Status Register (0x00690010, 0x00790010 / 0x0)

The Device Control Register provides information about PCI Express device specific parameters.

Table 1-201 TLU Device Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:6				Reserved field
TP	5	rst_l	0x0	R	Transactions Pending - This bit when set indicates that a port has issued Non-Posted Requests which have not been completed.

Table 1-201 TLU Device Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	4:0				Reserved field

1.3.4.103 TLU Link Capabilities Register (0x00690018, 0x00790018 / 0x15C81)

The Link Capabilities Register identifies PCI Express link specific capabilities.

Table 1-202 TLU Link Capabilities Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
PORT	31:24	rst_l	0x0	R	Port Number
RESERVED	23:18				Reserved field
L1	17:15	rst_l	0x2	R	L1 Exit Latency 010b = 2 us to less than 4 us
L0S	14:12	rst_l	0x5	R	L0s Exit Latency 101b = 1 us to less than 2 us
ASPM	11:10	rst_l	0x3	R	Active State Power Management Support 00b = reserved 01b = L0s entry supported 10b = reserved 11b = L0s and L1 supported
WIDTH	9:4	rst_l	0x8	R	Maximum Link Width 001000b = x8
SPEED	3:0	rst_l	0x1	R	Maximum Link Speed 0001b = 2.5Gb/s

1.3.4.104 TLU Link Control Register (0x00690020, 0x00790020 / 0x0)

The Link Control Register controls PCI Express Link specific parameters.

Table 1-203 TLU Link Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:8				Reserved field

Table 1-203 TLU Link Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
EXTSYNC	7	rst_1	0x0	RW	Extended Sync - This bit when set forces the transmission of 4096 FTS ordered sets in the L0s state followed by a single SKP ordered set prior to entering the L0 state, and the transmission of 1024 TS1 ordered sets in the L1 state prior to entering the Recovery state.
CLOCK	6	rst_1	0x0	RW	Common Clock Configuration - This bit when set indicates that this component and the component at the opposite end of this Link are operating with a distributed common reference clock.
RETRAIN	5	rst_1	0x0	RW	Retrain Link - A write of 1 to this bit initiates Link retraining by directing the Physical Layer LTSSM to the Recovery state
DISABLE	4	rst_1	0x0	RW	Link Disable - This bit disables the Link when set to 1b.
RCB	3	rst_1	0x0	R	Read Completion Boundary 0b = 64 byte
RESERVED	2				Reserved field
ASPM	1:0	rst_1	0x0	RW	Active State Link PM Control 00b = Disabled 01b = L0s entry enabled 10b = L1 entry enabled 11b = L0s and L1 entry supported

1.3.4.105 TLU Link Status Register (0x00690028, 0x00790028 / 0x0)

The Link Status Register provides information about PCI Express Link specific parameters.

Table 1-204 TLU Link Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:13				Reserved field
CLOCK	12	rst_1	0x0	R	Slot Clock Configuration - This bit indicates that the component uses the same physical reference clock that the platform provides.
TRAIN	11	rst_1	0x0	R	Link Training - This indicates that Link training is in progress (Physical Layer LTTSM in Configuration or Recovery State) or that 1 was written to the Retrain Link bit but Link training has not yet begun.
ERROR	10	rst_1	0x0	R	Training Error
WIDTH	9:4	rst_1	0x0	R	Negotiated Link Width 000000b = link down 000001b = x1 000010b = x2 (not supported) 000100b = x4 001000b = x8 001100b = x12 (not supported) 010000b = x16 (not supported) 100000b = x32 (not supported)
SPEED	3:0	rst_1	0x0	R	Link Speed 0000b = link down 0001b = 2.5 Gb/s

1.3.4.106 TLU Slot Capabilities Register (0x00690030, 0x00790030 / 0x0)

The Slot Capabilities Register identifies PCI Express slot specific capabilities. Any write to this register will cause the Set_Slot_Power_Limit Message to be transmitted on the link.

Table 1-205 TLU Slot Capabilities Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:17				Reserved field
SPLS	16:15	rst_l	0x0	RW	Set Power Limit Scale - Specifies the scale used for the Slot Power Limit Value. 00b - 1.0x 01b - 0.1x 10b - 0.01x 11b - 0.001x
SPLV	14:7	rst_l	0x0	RW	Set Power Limit Value. - In combination with the Slot Power Limit Scale value, specifies the upper limit on power supplied by slot. Power Limit (Watts) calculated by multiplying the value in this field by the value in the Slot Power Limit Scale field.
RESERVED	6:0				Reserved field

1.3.4.107 TLU Uncorrectable Error Log Enable Register (0x00691000, 0x00791000 / 0x17F011)

The TLU Uncorrectable Error Log Enable Register controls which events will be logged in the TLU Logged Uncorrectable Error Status Register.

Table 1-206 TLU Uncorrectable Error Log Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:21				Reserved field
EN	20:0	por_l	0x17F0 11	RW	Log Enables

1.3.4.108 TLU Uncorrectable Error Interrupt Enable Register (0x00691008, 0x00791008 / 0x0)

The TLU Uncorrectable Error Interrupt Enable Register controls which events logged in the TLU Logged Uncorrectable Error Status Register will generate an interrupt

Table 1-207 TLU Uncorrectable Error Interrupt Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:53				Reserved field
EN_S	52:32	rst_l	0x0	RW	Secondary Interrupt Enables
RESERVED	31:21				Reserved field
EN_P	20:0	rst_l	0x0	RW	Primary Interrupt Enables

1.3.4.109 TLU Uncorrectable Error Interrupt Status Register (0x00691010, 0x00791010 / 0x0)

The TLU Uncorrectable Error Interrupt Status Register indicates which events logged in the TLU Uncorrectable Error Status Clear Register are enabled to generate interrupts by the TLU Uncorrectable Error Interrupt Enable Register

Table 1-208 TLU Uncorrectable Error Interrupt Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:53				Reserved field
ERR_S	52:32	rst_l	0x0	R	Secondary Interrupt Status
RESERVED	31:21				Reserved field
ERR_P	20:0	rst_l	0x0	R	Primary Interrupt Status

1.3.4.110 TLU Uncorrectable Error Status Clear Register (0x00691018, 0x00791018 / 0x0)

The TLU Uncorrectable Error Status Clear Register indicates that an uncorrectable error occurred. Primary errors include data logged with these errors. Secondary events do not have any data logged with them. All errors belong to the same error group which means that any primary error logged will cause any additional error to get logged as secondary.

Table 1-209 TLU Uncorrectable Error Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:53				Reserved field

Table 1-209 TLU Uncorrectable Error Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
UR_S	52	por_1	0x0	RW1C	Unsupported Request Secondary Error
RESERVED	51				Reserved field
MFP_S	50	por_1	0x0	RW1C	Malformed TLP Secondary Error
ROF_S	49	por_1	0x0	RW1C	Receiver Overflow Secondary Error
UC_S	48	por_1	0x0	RW1C	Unexpected Completion Secondary
CA_S	47	por_1	0x0	RW1C	Completer Abort Secondary Error
CTO_S	46	por_1	0x0	RW1C	Completion Timeout Secondary Error
FCP_S	45	por_1	0x0	RW1C	Flow Control Protocol Secondary Error
PP_S	44	por_1	0x0	RW1C	Poisoned TLP Secondary Error
RESERVED	43:37				Reserved field
DLP_S	36	por_1	0x0	RW1C	Data Link Protocol Secondary Error
RESERVED	35:33				Reserved field
TE_S	32	por_1	0x0	RW1C	Training Secondary Error
RESERVED	31:21				Reserved field
UR_P	20	por_1	0x0	RW1C	Unsupported Request Primary Error
RESERVED	19				Reserved field
MFP_P	18	por_1	0x0	RW1C	Malformed TLP Primary Error
ROF_P	17	por_1	0x0	RW1C	Receiver Overflow Primary Error
UC_P	16	por_1	0x0	RW1C	Unexpected Completion Primary
CA_P	15	por_1	0x0	RW1C	Completer Abort Primary Error

Table 1-209 TLU Uncorrectable Error Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
CTO_P	14	por_1	0x0	RW1C	Completion Timeout Primary Error
FCP_P	13	por_1	0x0	RW1C	Flow Control Protocol Primary Error
PP_P	12	por_1	0x0	RW1C	Poisoned TLP Primary Error
RESERVED	11:5				Reserved field
DLP_P	4	por_1	0x0	RW1C	Data Link Protocol Primary Error
RESERVED	3:1				Reserved field
TE_P	0	por_1	0x0	RW1C	Training Primary Error

1.3.4.111 TLU Uncorrectable Error Status Set Register (0x00691020, 0x00791020 / 0x0)

The Uncorrectable Error Status Set Register controls setting errors in the TLU Uncorrectable Error Status Clear Register for testing purposes.

Table 1-210 TLU Uncorrectable Error Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:53				Reserved field
UR_S	52	por_1	0x0	RW1S	Unsupported Request Secondary Error
RESERVED	51				Reserved field
MFP_S	50	por_1	0x0	RW1S	Malformed TLP Secondary Error
ROF_S	49	por_1	0x0	RW1S	Receiver Overflow Secondary Error
UC_S	48	por_1	0x0	RW1S	Unexpected Completion Secondary
CA_S	47	por_1	0x0	RW1S	Completer Abort Secondary Error
CTO_S	46	por_1	0x0	RW1S	Completion Timeout Secondary Error

Table 1-210 TLU Uncorrectable Error Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
FCP_S	45	por_1	0x0	RW1S	Flow Control Protocol Secondary Error
PP_S	44	por_1	0x0	RW1S	Poisoned TLP Secondary Error
RESERVED	43:37				Reserved field
DLP_S	36	por_1	0x0	RW1S	Data Link Protocol Secondary Error
RESERVED	35:33				Reserved field
TE_S	32	por_1	0x0	RW1S	Training Secondary Error
RESERVED	31:21				Reserved field
UR_P	20	por_1	0x0	RW1S	Unsupported Request Primary Error
RESERVED	19				Reserved field
MFP_P	18	por_1	0x0	RW1S	Malformed TLP Primary Error
ROF_P	17	por_1	0x0	RW1S	Receiver Overflow Primary Error
UC_P	16	por_1	0x0	RW1S	Unexpected Completion Primary
CA_P	15	por_1	0x0	RW1S	Completer Abort Primary Error
CTO_P	14	por_1	0x0	RW1S	Completion Timeout Primary Error
FCP_P	13	por_1	0x0	RW1S	Flow Control Protocol Primary Error
PP_P	12	por_1	0x0	RW1S	Poisoned TLP Primary Error
RESERVED	11:5				Reserved field
DLP_P	4	por_1	0x0	RW1S	Data Link Protocol Primary Error
RESERVED	3:1				Reserved field
TE_P	0	por_1	0x0	RW1S	Training Primary Error

1.3.4.112 TLU Receive Uncorrectable Error Header1 Log Register (0x00691028, 0x00791028 / 0x0)

The TLU Receive Uncorrectable Error Header1 Log Register holds the first eight bytes of the received header when a primary event in the TLU Logged Uncorrectable Error Status Register is set.

Table 1-211 TLU Receive Uncorrectable Error Header1 Log Register

Field	Bits	Reset Name	Reset Value	Type	Description
HDR	63:0		64'bx	RW	[63:56] - Header Byte 0 [55:48] - Header Byte 1 [47:40] - Header Byte 2 [39:32] - Header Byte 3 [31:24] - Header Byte 4 [23:16] - Header Byte 5 [15:8] - Header Byte 6 [7:0] - Header Byte 7

1.3.4.113 TLU Receive Uncorrectable Error Header2 Log Register (0x00691030, 0x00791030 / 0x0)

The TLU Receive Uncorrectable Error Header1 Log Register holds the second eight bytes of the received header when a primary event in the TLU Logged Uncorrectable Error Status Register is set.

Table 1-212 TLU Receive Uncorrectable Error Header2 Log Register

Field	Bits	Reset Name	Reset Value	Type	Description
HDR	63:0		64'bx	RW	[63:56] - Header Byte 8 [55:48] - Header Byte 9 [47:40] - Header Byte A [39:32] - Header Byte B [31:24] - Header Byte C or Undefined [23:16] - Header Byte D or Undefined [15:8] - Header Byte E or Undefined [7:0] - Header Byte F or Undefined

1.3.4.114 TLU Transmit Uncorrectable Error Header1 Log Register (0x00691038, 0x00791038 / 0x0)

The TLU Transmit Uncorrectable Error Header1 Log Register holds the first eight bytes of the original transmitted request header when a primary event in the TLU Logged Uncorrectable Error Status Register is set on a completion.

Table 1-213 TLU Transmit Uncorrectable Error Header1 Log Register

Field	Bits	Reset Name	Reset Value	Type	Description
HDR	63:0		64'bx	RW	[63:56] - Header Byte 0 [55:48] - Header Byte 1 [47:40] - Header Byte 2 [39:32] - Header Byte 3 [31:24] - Header Byte 4 [23:16] - Header Byte 5 [15:8] - Header Byte 6 [7:0] - Header Byte 7

1.3.4.115 TLU Transmit Uncorrectable Error Header2 Log Register (0x00691040, 0x00791040 / 0x0)

The TLU Transmit Uncorrectable Error Header1 Log Register holds the second eight bytes of the original transmitted request header when a primary event in the TLU Logged Uncorrectable Error Status Register is set on a completion.

Table 1-214 TLU Transmit Uncorrectable Error Header2 Log Register

Field	Bits	Reset Name	Reset Value	Type	Description
HDR	63:0		64'bx	RW	[63:56] - Header Byte 8 [55:48] - Header Byte 9 [47:40] - Header Byte A [39:32] - Header Byte B [31:24] - Header Byte C or Undefined [23:16] - Header Byte D or Undefined [15:8] - Header Byte E or Undefined [7:0] - Header Byte F or Undefined

1.3.4.116 TLU Correctable Error Log Enable Register (0x006A1000, 0x007A1000 / 0x11C1)

The TLU Correctable Error Log Enable Register controls which events will be logged in the TLU Logged Correctable Error Status Register.

Table 1-215 TLU Correctable Error Log Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:13				Reserved field
EN	12:0	por_1	0x11C1	RW	Log Enables

1.3.4.117 TLU Correctable Error Interrupt Enable Register (0x006A1008, 0x007A1008 / 0x0)

The TLU Correctable Error Interrupt Enable Register controls which events logged in the TLU Logged Correctable Error Status Register will generate an interrupt

Table 1-216 TLU Correctable Error Interrupt Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:45				Reserved field
EN_S	44:32	rst_1	0x0	RW	Secondary Interrupt Enables
RESERVED	31:13				Reserved field
EN_P	12:0	rst_1	0x0	RW	Primary Interrupt Enables

1.3.4.118 TLU Correctable Error Interrupt Status Register (0x006A1010, 0x007A1010 / 0x0)

The TLU Correctable Error Interrupt Status Register indicates which events logged in the TLU Correctable Error Status Clear Register are enabled to generate interrupts by the TLU Correctable Error Interrupt Enable Register

Table 1-217 TLU Correctable Error Interrupt Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:45				Reserved field
ERR_S	44:32	rst_1	0x0	R	Secondary Interrupt Status
RESERVED	31:13				Reserved field
ERR_P	12:0	rst_1	0x0	R	Primary Interrupt Status

1.3.4.119 TLU Correctable Error Status Clear Register (0x006A1018, 0x007A1018 / 0x0)

The TLU Correctable Error Status Clear Register indicates that a correctable error occurred. All errors belong to the same error group which means that any primary error logged will cause any additional error to get logged as secondary

Table 1-218 TLU Correctable Error Status Clear Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:45				Reserved field
RTO_S	44	por_1	0x0	RW1C	Replay Timeout Secondary Error
RESERVED	43:41				Reserved field
RNR_S	40	por_1	0x0	RW1C	REPLAY_NUM Rollover Secondary Error
BDP_S	39	por_1	0x0	RW1C	Bad DLLP Secondary Error
BTP_S	38	por_1	0x0	RW1C	Bad TLP Secondary Error
RESERVED	37:33				Reserved field
RE_S	32	por_1	0x0	RW1C	Receiver Secondary Error
RESERVED	31:13				Reserved field
RTO_P	12	por_1	0x0	RW1C	Replay Timeout Primary Error
RESERVED	11:9				Reserved field
RNR_P	8	por_1	0x0	RW1C	REPLAY_NUM Rollover Primary Error
BDP_P	7	por_1	0x0	RW1C	Bad DLLP Primary Error
BTP_P	6	por_1	0x0	RW1C	Bad TLP Primary Error
RESERVED	5:1				Reserved field
RE_P	0	por_1	0x0	RW1C	Receiver Primary Error

1.3.4.120 TLU Correctable Error Status Set Register (0x006A1020, 0x007A1020 / 0x0)

The TLU Correctable Error Status Set Register controls setting errors in the TLU Correctable Error Status Clear Register for testing purposes.

Table 1-219 TLU Correctable Error Status Set Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:45				Reserved field
RTO_S	44	por_1	0x0	RW1S	Replay Timeout Secondary Error
RESERVED	43:41				Reserved field
RNR_S	40	por_1	0x0	RW1S	REPLAY_NUM Rollover Secondary Error
BDP_S	39	por_1	0x0	RW1S	Bad DLLP Secondary Error
BTP_S	38	por_1	0x0	RW1S	Bad TLP Secondary Error
RESERVED	37:33				Reserved field
RE_S	32	por_1	0x0	RW1S	Receiver Secondary Error
RESERVED	31:13				Reserved field
RTO_P	12	por_1	0x0	RW1S	Replay Timeout Primary Error
RESERVED	11:9				Reserved field
RNR_P	8	por_1	0x0	RW1S	REPLAY_NUM Rollover Primary Error
BDP_P	7	por_1	0x0	RW1S	Bad DLLP Primary Error
BTP_P	6	por_1	0x0	RW1S	Bad TLP Primary Error
RESERVED	5:1				Reserved field
RE_P	0	por_1	0x0	RW1S	Receiver Primary Error

1.3.4.121 LPU ID Register (0x006E2000, 0x007E2000 / 0x0)

This register contains information on the revision of the major modules, the serial lane width and the link byte width for the core.

Note: This register is Hardware Initialized.

Table 1-220 LPU ID Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:24				Reserved field
LTBWDTH	23:20	rst_l	4'bx	R	Link Type Byte Width 0x0 = Undefined 0x1 = 1-byte 0x2 = 2-byte 0x3 = 4-byte 0x4 = 8-byte 0x5 = 12-byte 0x6 = 16-byte 0x7 = 32-byte 0x8 - 0xf = reserved
PTLWDTH	19:16	rst_l	4'bx	R	PHY Type Lane Width 0x0 = Undefined, 0x1 = 1-lane 0x2 = 2-lane 0x3 = 4-lane 0x4 = 8-lane 0x5 = 12-lane 0x6 = 16-lane 0x7 = 32-lane 0x8 - 0xF = reserved
TRID	15:12	rst_l	4'bx	R	Transport Revision ID The transport revision number is 0x0
LNKID	11:8	rst_l	4'bx	R	Link Revision ID Hard-coded
PHYID	7:4	rst_l	4'bx	R	Phy Revision ID Hard-coded
GBID	3:0	rst_l	4'bx	R	SERDES Revision ID Hard-coded 0x1 - SERDES GFlx Rev 2.0

1.3.4.122 LPU Reset Register (0x006E2008, 0x007E2008 / 0x0)

This Reset Register provides a controllable soft reset for the modules within the Link, Physical, Physical Coding Sub-layer (PCS) and SERDES control inputs. This register is currently reset by the AHB reset `ur_ahb_rst` which is driven by `t2l_por`. The SERDES `ur_gbg_rst` is an asynchronous reset driven by `tl2_por`. The minimum pulse width of any reset is four (4) system clocks. Local WE is needed to write other fields of this register.

Note: Use these bits with caution or in diagnostic mode only. Only resetting these modules without resetting the rest of the chip may result in undefined behavior

The bits of this register should be 0 in normal operational mode.

This register will be updated to be reset by the master reset (`t2l_rst`) which must be OR'd with `t2l_por` externally.

Table 1-221 LPU Reset Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
RSTWE	31	<code>rst_l</code>	0x0	L	Reset Write Enable. When 1, allows bits 11:0 of this register to be written. Reads return 0.
RESERVED	30:12				Reserved field
RSTUNUSED	11:9	<code>rst_l</code>	0x0	RW	Unused Reset bits
RSTERROR	8	<code>rst_l</code>	0x0	RW	Reset Errors When 1, this resets the error status throughout the Link, Phy and PCS. This signal is OR's with power-on-reset signal <code>t2l_por</code> .
RSTTXLINK	7	<code>rst_l</code>	0x0	RW	Global Reset TXLINK When 1, resets the Txlink and control registers.
RSTRXLINK	6	<code>rst_l</code>	0x0	RW	Global Reset RXLINK When 1, resets the Rxlink and control registers.
RSTSMLINK	5	<code>rst_l</code>	0x0	RW	Global Reset SMLINK When 1, this resets the link and flow control state machines. This signal is OR'd with power-on-reset signal <code>t2l_por</code> .

Table 1-221 LPU Reset Register

Field	Bits	Reset Name	Reset Value	Type	Description
RSTLTSSM	4	rst_l	0x0	RW	Global Reset LTSSM When 1, resets the LTSSM, control and mask registers.
RSTTXPHY	3	rst_l	0x0	RW	Global Reset TXPHY
RSTRXPHY	2	rst_l	0x0	RW	Global Reset RXPHY
RSTTXPCS	1	rst_l	0x0	RW	Global Reset TXPCS When 1, the SERDES transmit reset is asserted and SERDES Glue transmit logic for all lanes are reset.
RSTRXPCS	0	rst_l	0x0	RW	Global Reset RXPCS When 1, the SERDES receiver reset is asserted, the SERDES Glue receive logic for the lane is reset and bits [15:0] in receiver phy config register is reset. Asserting this bit will not clear some registers within the PCS if the receiver PLL is disabled by power-down or receiver termination disable controls. See transmit phy status 2 and SERDES glue power down 2 register. Note for simulation: in fast sim mode, use this bit after setting new value in LPU LTSSM Config2 Register.

1.3.4.123 LPU Debug Status Register (0x006E2010, 0x007E2010 / 0x0)

This is a debug status register. It's for test purpose only.

Table 1-222 LPU Debug Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:16				Reserved field

Table 1-222 LPU Debug Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
DEBUGB	15:8	rst_l	0x0	R	debug B
DEBUGA	7:0	rst_l	0x0	R	debug A

1.3.4.124 LPU Debug Config Register (0x006E2018, 0x007E2018 / 0x0)

This is debug select register. Non zero values of debug B dbugb_blk_sel[31:25] & debug A dbuga_blk_sel[15:9] are reserved.

Table 1-223 LPU Debug Config Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
DBUGB_BLK_SEL	31:24	rst_l	0x0	RW	Debug B block select. This field selects the block in the LPU to monitor. Decode is 0: lu block 1: pu block
DBUGB_SIG_SEL	23:16	rst_l	0x0	RW	Debug B signal select. This field selects the 8-bit group of signals in debug B status register within the lpu
DBUGA_BLK_SEL	15:8	rst_l	0x0	RW	Debug A block select. This field selects the block in the LPU to monitor. Decode is 0: lu block 1: pu block
DBUGA_SIG_SEL	7:0	rst_l	0x0	RW	Debug A signal select. This field selects the 8-bit group of signals in debug A status register within the lpu.

1.3.4.125 LPU LTSSM Control Register (0x006E2020, 0x007E2020 / 0x0)

These bits are set by a processor write and cleared by hardware when the state is entered.

NOTE: RW1S bits should only be used for test purpose with cautions.

Reads of RW1S bits in this register return 0.

Bits[11:0] can be written into only if bit[31] in this register is set. Only 1 bit should be set at a time. Behaviour depends on LTSSM state when 2 bits are set at a time.

Table 1-224 LPU LTSSM Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
WR_ENABLE	31	rst_1	0x0	L	Write Enable
RESERVED	30:12				Reserved field
RCOVER_TO_CONFIG	11	rst_1	0x0	RW1S	Recovery to Configuration When 1 and LTSSM is in Recovery.idle, LTSSM will change to Configuration state to reconfigure the Link (to smaller width). Use Go_To_Detect.Quiet bit to reconfigure the link to the largest width. Reset by hardware when LTSSM is in the CONFIG_RCVRCFG or CONFIG_IDLE state. Note: set simultaneously with [10] to force entering Configuration state.
L0_TO_RECOVER	10	rst_1	0x0	RW1S	L0 to Recovery When 1 and in the L0 LTSSM state, will go to LTSSM Recovery state. Reset by hardware when LTSSM is in RECOVERY_RCVRLock or RECOVERY_RCVRCFG or RECOVERY_IDLE state.
UNUSED_0	9	rst_1	0x0	R	Unused 0
GO_TO_DETECT	8	rst_1	0x0	RW1S	Go to Detect.Quiet Request any state to Detect.Quiet state. This will also allow the link width to increase to the largest width possible.
UNUSED_1	7:4	rst_1	0x0	R	Unused 1

Table 1-224 LPU LTSSM Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
DISABLE_SCRAMBLING	3	rst_1	0x0	RW1S	<p>Disable Scrambling</p> <p>Writing a 1 to this field will force bit [27] of LPU LTSSM Config4 Register to be asserted. This will cause the Disable Scrambling bit to be set in the Training Control symbol of the TS1/TS2 sequence when TS1/TS2 are next sent.</p>
LINK_LOOPBK_REQ	2	rst_1	0x0	RW1S	<p>Link Loopback Request</p> <p>When 1 and in the Recovery or Configuration LTSSM states, LTSSM will become the loopback master and go to LOOPBACK_ENTRY or LOOPBACK_ACTIVE state. Reset by hardware when LTSSM is in LOOPBACK_ENTRY or LOOPBACK_ACTIVE state.</p> <p>Note: This bit is used for LTSSM debug control only. The core does not support Loopback Master mode.</p>
LINK_DISABLE_REQ	1	rst_1	0x0	RW	<p>Link Disable Request</p> <p>When 1 and in the Recovery or Configuration LTSSM states, LTSSM will go to DISABLED state.</p> <p>The link will remain disabled until the bit is cleared by a write to zero. This mode can not be used if a local processor is not involved. A PCI-Express write of this bit will hang the port until a power-on reset, soft reset or an AHB bus writes this bit to zero.</p>

Table 1-224 LPU LTSSM Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
HOT_RESET	0	rst_l	0x0	RW1S	Hot Reset request When 1 and in the Recovery LTSSM state, will go to LTSSM HOT_RESET state. Reset by hardware when the LTSSM is in the HOT_RESET state

1.3.4.126 LPU Link Status Register (0x006E2028, 0x007E2028 / 0x1)

Changes to this register will be propagated to the TLU Link Status Register.

Table 1-225 LPU Link Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:13				Reserved field
SLOT_CLK_CONFIG_PIN	12	rst_l	1'bx	R	Slot Clock Configuration - HW initialized When set, indicates that the downstream port is using the same reference clock as the downstream integrated device or the slot. The value of this field is changed by writing to the CONFIG[1] field in the TLU Control Register (tlu_ctl). Interface signal t2l_config[1]
LINK_TRAINING	11	rst_l	0x0	R	Link Training in Progress
LINK_TRAINING_ERR	10	rst_l	0x0	R	Link Training Error Once set, cleared by hardware when Link Training occurs successfully

Table 1-225 LPU Link Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
NEGOTIATED_WIDTH	9:4	rst_l	6'bx	R	Negotiated width 000000 Reserved 000001 X1 000010 X2 (not supported) 000100 X4 001000 X8 001100 X12 (not supported) 010000 X16 (default) 100000 X32 (not supported) Note: it's invalid before link is up
LINK_SPEED	3:0	rst_l	0x1	R	Link Speed 0001 2.5Gbs (default)

1.3.4.127 LPU Interrupt Status Register (0x006E2040, 0x007E2040 / 0x0)

This register indicates whether other Interrupt Status Registers in the LPU has an interrupt pending. The exceptions are the Performance Counter Overflows bits, which are actual events. Each implemented interrupt can be asserted by writing to the source interrupt test register with the interrupt mask equal to zero for that bit and the global bit within the register.

Note: This register is similar to Sun's domain core and block interrupt status register.

Table 1-226 LPU Interrupt Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
INTERRUPT	31	rst_l	0x0	R	Any Interrupt Active When 1, one of the unmasked interrupts below is active
RESERVED	30:8				Reserved field
INT_PERF_CNTR_2_OVF LW	7	rst_l	0x0	R	Performance Counter2 Overflow interrupt. This bit is cleared using the LPU Link Performance Counter Control Register (pcie_lpu_link_perf_cntr_ctl)

Table 1-226 LPU Interrupt Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
INT_PERF_CNTR_1_OVF LW	6	rst_l	0x0	R	Performance Counter1 Overflow interrupt. This bit is cleared using the LPU Link Performance Counter Control Register (pcie_lpu_link_perf_cntr_ctl)
INT_LINK_LAYER	5	rst_l	0x0	R	LPU Link Layer Interrupt Pending See event(s) logged in the LPU Link Layer Interrupt and Status Register (pcie_lpu_ll_err_int)
INT_PHY_ERROR	4	rst_l	0x0	R	LPU PHY Error Interrupt Pending See event(s) logged in the LPU Phy Layer Interrupt and Status Interrupt Register (pcie_lpu_phy_err_int)
INT_LTSSM	3	rst_l	0x0	R	LPU LTSSM Interrupt Pending See event(s) logged in the LPU LTSSM Interrupt and Status Register (pcie_lpu_ltssm_int)
INT_PHY_TX	2	rst_l	0x0	R	LPU PHY Tx Interrupt Pending See event(s) logged in the LPU Transmit Phy Interrupt and Status Register (pcie_lpu_tx_phy_int)
INT_PHY_RX	1	rst_l	0x0	R	LPU PHY Rx Interrupt Pending See event(s) logged in the LPU Receive Phy Interrupt and Status Register (pcie_lpu_rx_phy_int)
INT_PHY_GB	0	rst_l	0x0	R	LPU PHY GB Interrupt Pending See event(s) logged in the LPU SERDES Glue Interrupt and Status Test Register (pcie_lpu_gb_gl_int)

1.3.4.128 LPU Interrupt Mask Register (0x006E2048, 0x007E2048 / 0x800000FF)

This is LPU interrupt mask register, which is similar to Sun's core and block interrupt enable register.

Note: write 0 to enable interrupt and 1 to disable interrupt.

Table 1-227 LPU Interrupt Mask Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
MSK_INTERRUPT_EN	31	rst_1	0x1	RW	LPU Interrupt Enable Mask Note: set it to zero to allow perf cntr intrrupt
RESERVED	30:8				Reserved field
MSK_PERF_CNTR_2_OVF LW	7	rst_1	0x1	RW	Performance Counter2 Overflow Mask Note: set it to zero in normal oper- ation
MSK_PERF_CNTR_1_OVF LW	6	rst_1	0x1	RW	Performance Counter1 Overflow Mask Note: set it to zero in normal oper- ation
MSK_LINK_LAYER	5	rst_1	0x1	RW	Link Layer Interrupt Mask Note: used for simulation only. Leave it masked in normal opera- tion
MSK_PHY_ERROR	4	rst_1	0x1	RW	PHY Error Interrupt Mask Note: used for simulation only. Leave it masked in normal opera- tion
MSK_LTSSM	3	rst_1	0x1	RW	LTSSM Interrupt Mask Note: used for simulation only. Leave it masked in normal opera- tion
MSK_PHY_TX	2	rst_1	0x1	RW	PHY Tx Interrupt Mask Note: used for simulation only. Leave it masked in normal opera- tion
MSK_PHY_RX	1	rst_1	0x1	RW	PHY Rx Interrupt Mask Note: used for simulation only. Leave it masked in normal opera- tion

Table 1-227 LPU Interrupt Mask Register

Field	Bits	Reset Name	Reset Value	Type	Description
MSK_PHY_GB	0	rst_l	0x1	RW	PHY GB Interrupt Mask Note: used for simulation only. Leave it masked in normal operation

1.3.4.129 LPU Link Performance Counter Select Register (0x006E2100, 0x007E2100 / 0x0)

This is link performance counter select register. It provides the selects for the performance counters.

Table 1-228 LPU Link Performance Counter Select Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
PERF_CNTR2_SELECT	31:16	rst_l	0x0	RW	Performance Counter2 Select The values for Select 2 are the same as select 1

Table 1-228 LPU Link Performance Counter Select Register

Field	Bits	Reset Name	Reset Value	Type	Description
PERF_CNTR1_SELECT	15:0	rst_l	0x0	RW	Performance Counter1 Select 0xFFFF None (counter retains previous value) 0xFFFE - 0x13 Reserved 0x12 Cycles for TLP, DLLP receive in progress 0x11 Cycles for TLP, DLLP transmit in progress 0x10 Clock cycles 0xF Cycles of TLP transmit in progress (include replay) 0xE cycles of Replay in progress 0xD LTSSM Recovery State Entries 0xC Receiver Errors 0xB ACK DLLPs Sent 0xA NAK DLLPs Sent 0x9 Bad TLPs 0x8 Bad DLLPs 0x7 ACK/NAK Latency Timer Timeouts 0x6 Replay Timer Timeouts 0x5 Retries Started 0x4 NAK DLLPs Received 0x3 ACK DLLPs Received 0x2 DLLPs Received 0x1 TLPs received 0x0 Reset (reset counter to 0)

1.3.4.130 LPU Link Performance Counter Control Register (0x006E2110, 0x007E2110 / 0x0)

Reads of RW1S and RW1C bits in this register return 0. RW1C bit overrides RW1S bit if the two corresponding bits are written to 1.

Table 1-229 LPU Link Performance Counter Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:7				Reserved field

Table 1-229 LPU Link Performance Counter Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
SET_PERF_CNTR2_OVERFLOW	6	rst_l	0x0	RW1S	Set Performance counter2 overflow Interrupt bit in the LPU Interrupt Status Register
SET_PERF_CNTR1_OVERFLOW	5	rst_l	0x0	RW1S	Set Performance counter1 overflow Interrupt bit in the LPU Interrupt Status Register
RESERVED	4				Reserved field
RST_PERF_CNTR2_OVERFLOW	3	rst_l	0x0	RW1C	Reset Performance Counter2 overflow Interrupt bit in the LPU Interrupt Status Register
RST_PERF_CNTR2	2	rst_l	0x0	RW1C	Reset Performance Counter2 value
RST_PERF_CNTR1_OVERFLOW	1	rst_l	0x0	RW1C	Reset Performance Counter1 overflow Interrupt bit in the LPU Interrupt Status Register
RST_PERF_CNTR1	0	rst_l	0x0	RW1C	Reset Performance Counter1 value

1.3.4.131 LPU Link Performance Counter1 (0x006E2120, 0x007E2120 / 0x0)

This is perf. counter 1 value register.

Table 1-230 LPU Link Performance Counter1

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
PERF_CNTR1	31:0	por_l	0x0	R	Performance Counter1 Value

1.3.4.132 LPU Link Performance Counter1 Test (0x006E2128, 0x007E2128 / 0x0)

Writing a value to this register loads the value in LPU Link Performance Counter1 Register.

Table 1-231 LPU Link Performance Counter1 Test

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field

Table 1-231 LPU Link Performance Counter1 Test

Field	Bits	Reset Name	Reset Value	Type	Description
PERF_CNTR1_TEST	31:0	rst_l	0x0	L	Performance Counter1 Test Writing a value to this register loads the value in performance counter1. This register always returns 0.

1.3.4.133 LPU Link Performance Counter2 (0x006E2130, 0x007E2130 / 0x0)

This is perf. counter 2 value register.

Table 1-232 LPU Link Performance Counter2

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
PERF_CNTR2	31:0	por_l	0x0	R	Performance Counter2 Value

1.3.4.134 LPU Link Performance Counter2 Test (0x006E2138, 0x007E2138 / 0x0)

A value written to register is loaded to LPU Link Performance Counter2 Register.

Table 1-233 LPU Link Performance Counter2 Test

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
PERF_CNTR2_TEST	31:0	rst_l	0x0	L	Performance Counter2 Test Writing a value to this register loads the value in performance counter2. This register always returns 0.

1.3.4.135 LPU Link Layer Config Register (0x006E2200, 0x007E2200 / 0x100)

To use auto-update threshold feature, set

bit[4] to 1'b1;

bit[16] to 1'b0;

bit[19] to 1'b0.

To manually adjust threshold values, set

bit[16] to 1'b1;

bit[19] to 1'b1.

Note:

(1) set bit[17] to 1'b1 and bit[18] to 1'b0 in normal operation.

(2) set bit[18] to 1'b1 only for diagnostic purpose.

Table 1-234 LPU Link Layer Config Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:20				Reserved field
AUTO_UPDATE_DIS	19	rst_1	0x0	RW	Automatic Update Disable When set it disables automatic hardware update of Nak_LATENCY_TIMER threshold and Replay_Timer threshold.
FREQ_NAK_EN	18	rst_1	0x0	RW	Frequent Nak enable This bit allows duplicate Naks to be sent out if no good TLP has been received after sending out first Nak and Ack Nak latency timer is greater than Freq Nak Latency Timer threshold. This bit is set for test purpose only.
REPLAY_AFTER_REC	17	rst_1	0x0	RW	Replay after Recovery Enable When this bit is set Replay from retry buffer is enabled after Recovery state exit to L0 state.
LAT_THRES_WR_EN	16	rst_1	0x0	RW	Latency threshold write enable When set enables writes to following registers: Nak_LATENCY_TIMER threshold and Replay_Timer threshold. It is recommended to set Automatic Update Disable bit [19] while writing a 1'b1 to Latency threshold write enable. The Automatic Update Disable bit [19] should be at 1'b1 in order to retain the values written in Latency threshold registers.

Table 1-234 LPU Link Layer Config Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	15:9				Reserved field
VC0_EN	8	rst_1	0x1	RW	Virtual Channel 0 (VC0) Enable This bit is set to 1 by default. It should remain 1 for normal operation.
UNUSED	7:5	rst_1	0x0	RW	Unused
L0S_ADJ_FAC_EN	4	rst_1	0x0	RW	L0s adjustment factor enable - effective only when threshold auto-update is enabled When it's 1'b1: Replay_Timer threshold value get automatically updated with Rx_L0s adjustment factor; Nak_LATENCY_TIMER thresh- old get automatically updated with Tx_L0s adjustment factor if t2l_link_control[0] is 1'b1, i.e. L0s power management state is enabled. When it's 1'b0: Replay_Timer and LATENCY threshold values get automatically updated without Rx_L0s/Tx_L0s adjustment factors if threshold auto-update is enabled.
TLP_XMIT_FC_EN	3	rst_1	0x0	RW	TLP xmit FC_INIT2 enable This bit allows TLPs to be sent out in FC_INIT2 state of VC0. This bit is reset to zero by default.
FREQ_ACK_ENABLE	2	rst_1	0x0	RW	Frequent Ack Enable This bit allows Acks to be sent on Ack Nak Latency timeout even if All received TLP's have been acknowledged.

Table 1-234 LPU Link Layer Config Register

Field	Bits	Reset Name	Reset Value	Type	Description
RETRY_DISABLE	1	rst_l	0x0	RW	Retry disable This bit is used in debug mode only when Firmware reads and writes retry buffer.
RESERVED	0				Reserved field

1.3.4.136 LPU Link Layer Status Register (0x006E2208, 0x007E2208 / 0x1)

Local WE is needed to write corresponding field of this register. Write to this register is for test purpose only.

Table 1-235 LPU Link Layer Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:10				Reserved field
INIT_FC_SM_WE	9	rst_l	0x0	L	Init Flow Control State Machine Write Enable. Always returns 0.
LNK_ST_DLUP_WE	8	rst_l	0x0	L	Link State / DL up Write Enable. Always returns 0.
RESERVED	7:6				Reserved field
INIT_FC_SM_STS	5:4	rst_l	0x0	RW	Init Flow Control State Machine Status 00 FC_Idle 01 FC_INIT1 11 FC_INIT2 10 FC_INIT DONE
DLUP_STS	3	rst_l	0x0	RW	DL up Status
LNK_STATE_MACH_STS	2:0	rst_l	0x1	RW	Link State Machine Status 001 DL_Inactive 010 DL_Init 100 DL_Active

1.3.4.137 LPU Link Layer Interrupt and Status Register (0x006E2210, 0x007E2210/0x0)

This register is similar to Sun's domain error status clear register except Bit [31], which is similar to Sun's domain interrupt status register.

Table 1-236 LPU Link Layer Interrupt and Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
INT_LINK_ERR_ACT	31	por_1	0x0	R	Any Link Error active interrupt It's 1 when at least one of the unmasked status bits are asserted and the corresponding bit is unmasked in interrupt mask register.
RESERVED	30:23				Reserved field
INT_UNSPRTD_DLLP	22	por_1	0x0	RW1C	Unsupported DLLP
INT_DLLP_RCV_ERR	21	por_1	0x0	RW1C	DLLP with Receive Error Note: Don't use this bit, for simulation only.
INT_BAD_DLLP	20	por_1	0x0	RW1C	Bad DLLP
RESERVED	19				Reserved field
INT_TLP_RCV_ERR	18	por_1	0x0	RW1C	TLP with Receive Error Note: Don't use this bit, for simulation only.
INT_SRC_ERR_TLP	17	por_1	0x0	RW1C	Source Error TLP (TLP with inverted CRC and EDB)
INT_BAD_TLP	16	por_1	0x0	RW1C	Bad TLP
RESERVED	15:10				Reserved field
INT_RTRY_BUF_UDF_ERR	9	por_1	0x0	RW1C	Retry Buffer Underflow Error
INT_RTRY_BUF_OVF_ERR	8	por_1	0x0	RW1C	Retry Buffer Overflow Error
INT_EG_TLP_MIN_ERR	7	por_1	0x0	RW1C	Egress TLP Minimum Length Error

Table 1-236 LPU Link Layer Interrupt and Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
INT_EG_TRNC_FRM_ERR	6	por_1	0x0	RW1C	Egress Truncated Frame Error
INT_RTRY_BUF_PE	5	por_1	0x0	RW1C	Retry Buffer Parity Error
INT_EGRESS_PE	4	por_1	0x0	RW1C	Egress Parity Error
RESERVED	3				Reserved field
INT_RPLAY_TMR_TO	2	por_1	0x0	RW1C	Replay Timer Time Out Note: Use TLU copy in CE registers
INT_RPLAY_NUM_RO	1	por_1	0x0	RW1C	Replay Number Rollover Note: Use TLU copy in CE registers
INT_DLNK_PES	0	por_1	0x0	RW1C	Data Link Protocol Error Status

1.3.4.138 LPU Link Layer Interrupt and Status Test Register (0x006E2218, 0x007E2218 / 0x0)

This is an error simulation register for the LPU Link Layer Interrupt and Status Register. This register is similar to Sun's domain error status set register.

Note: Writing 1 to each bit sets corresponding bit in Link Layer Error status clear Register. Reads of this register return 0.

Table 1-237 LPU Link Layer Interrupt and Status Test Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:23				Reserved field
TST_UNSPRTD_DLLP	22	rst_1	0x0	RW1S	Unsupported DLLP Test
TST_DLLP_RCV_ERR	21	rst_1	0x0	RW1S	DLLP with Receive Error Test
TST_BAD_DLLP	20	rst_1	0x0	RW1S	Bad DLLP Test
RESERVED	19				Reserved field
TST_TLP_RCV_ERR	18	rst_1	0x0	RW1S	TLP with Receive Error Test
TST_SRC_ERR_TLP	17	rst_1	0x0	RW1S	Source Error TLP Test
TST_BAD_TLP	16	rst_1	0x0	RW1S	Bad TLP Test
RESERVED	15:10				Reserved field

Table 1-237 LPU Link Layer Interrupt and Status Test Register

Field	Bits	Reset Name	Reset Value	Type	Description
TST_RTRY_BUF_UDF_ERR	9	rst_l	0x0	RW1S	Retry Buffer Underflow Error test
TST_RTRY_BUF_OVF	8	rst_l	0x0	RW1S	Retry Buffer Overflow error Test
TST_EG_TLP_MIN_ERR	7	rst_l	0x0	RW1S	Egress TLP Minimum Length Error Test
TST_EG_TRNC_FRM_ERR	6	rst_l	0x0	RW1S	Egress Truncated Frame Error Test
TST_RTRY_BUF_PE	5	rst_l	0x0	RW1S	Retry Buffer Parity Error Test
TST_EGRESS_PE	4	rst_l	0x0	RW1S	Egress Parity Error Test
RESERVED	3				Reserved field
TST_RPLAY_TMR_TO	2	rst_l	0x0	RW1S	Replay Timer Time Out Test
TST_RPLAY_NUM_RO	1	rst_l	0x0	RW1S	Replay Number Rollover Test
TST_DLNK_PES	0	rst_l	0x0	RW1S	Data Link Protocol Error Status Test

1.3.4.139 LPU Link Layer Interrupt Mask Register (0x006E2220, 0x007E2220 / 0x807FFFFF)

Write 0 to enable interrupt and 1 to disable interrupt.

Table 1-238 LPU Link Layer Interrupt Mask Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
MSK_LINK_ERR_ACT	31	rst_l	0x1	RW	Any Link Error Active Mask
RESERVED	30:23				Reserved field
MSK_UNSPRTD_DLLP	22	rst_l	0x1	RW	Unsupported DLLP Mask
MSK_DLLP_RCV_ERR	21	rst_l	0x1	RW	DLLP with Receive Error Mask
MSK_BAD_DLLP	20	rst_l	0x1	RW	Bad DLLP Mask
MSK_UNUSED_2	19	rst_l	0x1	RW	Unused
MSK_TLP_RCV_ERR	18	rst_l	0x1	RW	TLP with Receive Error Mask

Table 1-238 LPU Link Layer Interrupt Mask Register

Field	Bits	Reset Name	Reset Value	Type	Description
MSK_SRC_ERR_TLP	17	rst_l	0x1	RW	Source Error TLP Mask
MSK_BAD_TLP	16	rst_l	0x1	RW	Bad TLP Mask
MSK_UNUSED_1	15:10	rst_l	0x3F	RW	Unused
MSK_RTRY_UNF_OVF	9	rst_l	0x1	RW	Retry Buffer Underflow Mask
MSK_RTRY_BUF_OVF	8	rst_l	0x1	RW	Retry Buffer Overflow Mask
MSK_EG_TLP_MIN_ERR	7	rst_l	0x1	RW	Egress TLP Minimum Length Error Mask
MSK_EG_TRNC_FRM_ERR	6	rst_l	0x1	RW	Egress Truncated Frame Error Mask
MSK_RTRY_BUF_PE	5	rst_l	0x1	RW	Retry Buffer Parity Error Mask
MSK_EGRESS_PE	4	rst_l	0x1	RW	Egress Parity Error Mask
MSK_UNUSED_0	3	rst_l	0x1	RW	Unused
MSK_RPLAY_TMR_TO	2	rst_l	0x1	RW	Replay Timer Time Out Mask
MSK_RPLAY_NUM_RO	1	rst_l	0x1	RW	Replay Number Rollover Mask
MSK_DLNK_PES	0	rst_l	0x1	RW	Data Link Protocol Error Status Mask

1.3.4.140 LPU Flow Control Update Control Register (0x006E2240, 0x007E2240 / 0x3)

This is link layer FC update control register

Table 1-239 LPU Flow Control Update Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:3				Reserved field
FC0_U_C_EN	2	rst_l	0x0	RW	Flow Control 0 (FC0) Update completion enable
FC0_U_NP_EN	1	rst_l	0x1	RW	Flow Control 0 (FC0) Update non-posted enable

Table 1-239 LPU Flow Control Update Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
FC0_U_P_EN	0	rst_l	0x1	RW	Flow Control 0 (FC0) Update posted enable

1.3.4.141 LPU Link Layer Flow Control Update Timeout Value Register (0x006E2260, 0x007E2260 / 0x1D4C)

This is link layer FC update timeout value.

Table 1-240 LPU Link Layer Flow Control Update Timeout Value Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:15				Reserved field
FC_UPDATE_TO	14:0	rst_l	0x1D4C	RW	Flow Control resend timeout value The default is equivalent to 30 us with 4ns per count.

1.3.4.142 LPU Link Layer VC0 Flow Control Update Timer0 Register (0x006E2268, 0x007E2268 / 0x0)

This is Link Layer VC0 Flow Control Update Timer0 Register.

Table 1-241 LPU Link Layer VC0 Flow Control Update Timer0 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:31				Reserved field
VC0_FC_UP_TMR_NP	30:16	rst_l	0x0	R	VC0 Flow Control Update Timer value Non-Posted request
RESERVED	15				Reserved field
VC0_FC_UP_TMR_P	14:0	rst_l	0x0	R	VC0 Flow Control Update Timer value Posted request

1.3.4.143 LPU Link Layer VC0 Flow Control Update Timer1 Register (0x006E2270, 0x007E2270 / 0x0)

This is Link Layer VC0 Flow Control Update Timer1 Register.

Table 1-242 LPU Link Layer VC0 Flow Control Update Timer1 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:15				Reserved field
VC0_FC_UP_TMR_CPL	14:0	rst_l	0x0	R	VC0 Flow Control Update Timer value Completions

1.3.4.144 LPU Txlink Frequent Nak Latency Timer Threshold Register (0x006E2400, 0x007E2400 / 0x43)

This register takes effective only when frequent Nak is enabled, i.e., bit [18] is 1'b1 in Link Layer Config Register. This register provides the threshold count between sending two consecutive Naks with same sequence number when Frequent Nak is enabled. This register can be written into when Latency timer Write Enable [16] is asserted in Link Layer Config Register, else it retains its old value or the value updated by hardware. Hardware update can be disabled by writing a 1'b1 to Automatic Update Disable bit [19] in Link Layer Config Register. The value to program into this register is a function of the configured link width, max payload size, and Tx_L0s adjustment factor. In auto-update, Tx_L0s adjustment factor is taken into account only when bit[4] in LPU Link Layer Config Register is 1'b1 and bit[0] in TLU Link Capabilities Register is 1'b1.

Note: To config this register, refer to values in the Frequent Nak Latency Threshold table in section of Link Layer Thresholds.

Table 1-243 LPU Txlink Frequent Nak Latency Timer Threshold Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:16				Reserved field
ACK_NAK_THR	15:0	rst_l	0x43	RW	Frequent Nak Latency Timer Threshold

1.3.4.145 LPU Txlink AckNak Latency Timer Register (0x006E2408, 0x007E2408 / 0x0)

This is a count up register. Frequent Nak Latency timeout happens when the value in this register is equal to the value in Frequent Nak Latency Threshold Register. High priority Ack is scheduled when the timer value is greater than Ack Latency threshold.

Table 1-244 LPU Txlink AckNak Latency Timer Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:16				Reserved field
ACK_NAK_TMR	15:0	rst_l	0x0	R	Ack Nak Latency Timer

1.3.4.146 LPU Txlink Replay Timer Threshold Register (0x006E2410, 0x007E2410 / 0xFC)

This register can be written into when Latency timer Write Enable bit [16] is asserted in Link Layer Config Register, else it retains its old value or the value updated by hardware. Hardware update can be disabled by writing a 1'b1 to Automatic Update Disable bit [19] in Link Layer Config Register. The value to program into this register is a function of the configured link width, max payload size, and Rx_L0s adjustment factor. In auto-update, Rx_L0s adjustment factor is taken into account only when bit[4] in LPU Link Layer Config Register is 1'b1.

Note: To config this register manually, refer to values in the Replay Timer Threshold table in section of Link Layer Thresholds.

Table 1-245 LPU Txlink Replay Timer Threshold Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:20				Reserved field
RPLAY_TMR_THR	19:0	rst_l	0xFC	RW	Replay Timer Threshold

1.3.4.147 LPU Txlink Replay Timer Register (0x006E2418, 0x007E2418 / 0xFC)

This is Txlink Replay Timer Register

Table 1-246 LPU Txlink Replay Timer Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:20				Reserved field

Table 1-246 LPU Txlink Replay Timer Register

Field	Bits	Reset Name	Reset Value	Type	Description
RPLAY_TMR	19:0	rst_l	0xFC	R	Replay Timer This is a count down register. Replay timeout happens when the value in this register is 0. It gets loaded with value in Replay timer threshold register whenever Replay timer restarts.

1.3.4.148 LPU Txlink Replay Number Status Register (0x006E2420, 0x007E2420 / 0x0)

Local WE needed to write other fields of this register. Write to this register for test purpose only.

Table 1-247 LPU Txlink Replay Number Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
WE	31	rst_l	0x0	RW	Write Enable
RESERVED	30:2				Reserved field
RPLAY_NUM_CNTR	1:0	rst_l	0x0	RW	Replay Number Counter This field is read only and write for test purpose with write enable.

1.3.4.149 LPU Replay Buffer Max Address Register (0x006E2428, 0x007E2428 / 0xB3F)

This register should not be changed during normal operation (after link training).

Table 1-248 LPU Replay Buffer Max Address Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:16				Reserved field

Table 1-248 LPU Replay Buffer Max Address Register

Field	Bits	Reset Name	Reset Value	Type	Description
RTRY_BUFF_MAX_ADDR	15:0	rst_l	0xB3F	RW	<p>Retry Buffer Max Address</p> <p>This field contains the retry buffer maximum address location. It should be a value 8N-1, where N is a positive integer. The value in this field can be changed only while link is in DL_Inactive state. If the value is changed in DL_Active state, then the link needs to be brought down by driving LTSSM to detect state.</p>

1.3.4.150 LPU Txlink Retry FIFO Pointer Register (0x006E2430, 0x007E2430 / 0xFFFF0000)

Write to this register for test purpose only.

Table 1-249 LPU Txlink Retry FIFO Pointer Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
RTRY_FIFO_TLPTR	31:16	rst_l	0xFFFF	RW	<p>Retry FIFO Tail Pointer</p> <p>This field can only be written if the retry_disable bit in the pcie_lpu_ll_config register is 1.</p>
RTRY_FIFO_HDPTR	15:0	rst_l	0x0	RW	<p>Retry FIFO Head Pointer</p> <p>This field can only be written if the retry_disable bit in the pcie_lpu_ll_config register is 1.</p>

1.3.4.151 LPU Txlink Retry FIFO R/W Pointer Register (0x006E2438, 0x007E2438 / 0x0)

For debug and lab use only.

Table 1-250 LPU Txlink Retry FIFO R/W Pointer Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
RTRY_BFFR_WRPTR	31:16	rst_l	0x0	R	Retry Buffer Write Pointer

Table 1-250 LPU Txlink Retry FIFO R/W Pointer Register

Field	Bits	Reset Name	Reset Value	Type	Description
RTRY_BFFR_RDPTR	15:0	rst_l	0x0	R	Retry Buffer Read Pointer

1.3.4.152 LPU Txlink Retry FIFO Credit Register (0x006E2440, 0x007E2440 / 0xB40)

For debug and lab use only.

Table 1-251 LPU Txlink Retry FIFO Credit Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:16				Reserved field
RTRY_FIFO_CRDT	15:0	rst_l	0xB40	R	Retry FIFO Credit

1.3.4.153 LPU Txlink Sequence Counter Register (0x006E2448, 0x007E2448 / 0xFFF0000)

Local WE needed to write certain fields of this register. Write to this register for test purpose only.

Table 1-252 LPU Txlink Sequence Counter Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
WE	31	rst_l	0x0	L	Write Enable for [11:0]
ACK_SEQ_WE	30	rst_l	0x0	L	Ack Seq Write Enable for [27:16]
RESERVED	29:28				Reserved field
ACK_SEQ_CNTR	27:16	rst_l	0xFFF	RW	Ackd sequence counter
RESERVED	15:12				Reserved field
NXT_TX_SEQ_CNTR	11:0	rst_l	0x0	RW	Next Transmit Sequence Counter

1.3.4.154 LPU Txlink Ack Sent Sequence Number Register (0x006E2450, 0x007E2450 / 0xFFFF)

For debug and lab use only.

Table 1-253 LPU Txlink Ack Send Sequence Number Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:12				Reserved field
SEQ_NUM	11:0	rst_l	0xFFFF	R	Sequence Number in last Ack/Nak sent

1.3.4.155 LPU Txlink Sequence Count FIFO Max Addr Register (0x006E2458, 0x007E2458 / 0x167)

For debug and lab use only.

Table 1-254 LPU Txlink Sequence Count FIFO Max Addr Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:12				Reserved field
SEQ_CNT_MAX_ADDR	11:0	rst_l	0x167	RW	Sequence Count FIFO Max Address This field contains the sequence buffer maximum address location. The value in this field can be changed only while link is in DL_Inactive state. If the value is changed in DL_Active state, then the link needs to be brought down by driving LTSSM to detect state.

1.3.4.156 LPU Txlink Sequence Count FIFO Pointers Register (0x006E2460, 0x007E2460 / 0xFFFF0000)

Write to this register for test purpose only.

Table 1-255 LPU Txlink Sequence Count FIFO Pointers Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:28				Reserved field

Table 1-255 LPU Txlink Sequence Count FIFO Pointers Register

Field	Bits	Reset Name	Reset Value	Type	Description
SEQ_CNT_TLPTR	27:16	rst_l	0xFFFF	RW	Sequence Count Tail Pointer Writes take effect only if retry_disable bit in pcie_lpu_ll_config register is 1.
RESERVED	15:12				Reserved field
SEQ_CNT_HDPTR	11:0	rst_l	0x0	RW	Sequence Count Head Pointer Writes take effect only if retry_disable bit in pcie_lpu_ll_config register is 1.

1.3.4.157 LPU Txlink Sequence Count R/W Pointers Register (0x006E2468, 0x007E2468 / 0x0)

This is Txlink Sequence Count R/W Pointers Register.

Table 1-256 LPU Txlink Sequence Count R/W Pointers Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:28				Reserved field
SEQ_CNT_WRPTR	27:16	rst_l	0x0	R	Sequence Count Write Pointer
RESERVED	15:12				Reserved field
SEQ_CNT_RDPTR	11:0	rst_l	0x0	R	Sequence Count Read Pointer

1.3.4.158 LPU Txlink Test Control Register (0x006E2470, 0x007E2470 / 0x0)

This is an error injection register. Writing a 1 initiates the corresponding action on the transmit side, affecting a TLP or DLLP. Reads of RW1S bits return 0. These test bits should be set only after link is up, i.e. bit [3] in link layer status register is asserted. Write to this register for test purpose only

Table 1-257 LPU Txlink Test Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:4				Reserved field
DIS_ACK	3	rst_l	0x0	RW	Disable Ack

Table 1-257 LPU Txlink Test Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
FORCE_NAK	2	rst_l	0x0	RW1S	Force sending of a NAK DLLP This test feature should only be used in a controlled environment as of that one or more TLPs are on the way to Fire from a remote device connected to Fire. Probably don't want to force Nak's on Idle cycles.
FORCE_BAD_TLP_CRC	1	rst_l	0x0	RW1S	Force bad (inverted) CRC during transmission of TLP
FORCE_RTX_TLP	0	rst_l	0x0	RW1S	Force retransmit of TLP This test feature can only be used in a controlled environment as of: 1) retry buffer is NOT empty and 2) no Acks or Naks are being received and 3) no packets are being transmitted and 4) replay timer is not timing out.

1.3.4.159 LPU Txlink Memory Address Control Register (0x006E2480, 0x007E2480 / 0x0)

This register is used to read and write from retry fifo and sequence fifo in test mode only. It's recommended to assert disable retry bit in link layer config register before using this register. Disable retry should be deasserted after using this register. Reads of RW1S bits return 0.

Table 1-258 LPU Txlink Memory Address Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
DONE	31	rst_l	0x0	RW1C	Done asserted by hardware when memory read or write is done. Needs to be cleared by firmware.

Table 1-258 LPU Txlink Memory Address Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
GO_BIT	30	rst_1	0x0	RW1S	Go Bit Firmware should write to this bit to do memory read or write. Hardware automatically clears this bit when read or write is finished.
RD_WR_SEL	29	rst_1	0x0	RW	Read/Write Select 0 - Read 1 - Write
FIFO_SEL	28	rst_1	0x0	RW	FIFO Select 0 - Retry FIFO 1 - Sequence Count FIFO
RESERVED	27:16				Reserved field
MEM_ADDR	15:0	rst_1	0x0	RW	Memory Address

1.3.4.160 LPU Txlink Memory Data Load0 Register (0x006E2488, 0x007E2488 / 0x0)

This register stores the read/write data from sequence buffer/retry_buffer{byte3,byte2,byte1,byte0} when a sequence buffer/retry_buffer CPU access read/write takes place.

Table 1-259 LPU Txlink Memory Data Load0 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
MEM_RD_WR_DATA0	31:0	rst_1	0x0	RW	Memory Read/Write Data0

1.3.4.161 LPU Txlink Memory Data Load1 Register (0x006E2490, 0x007E2490 / 0x0)

This register stores the read/write data from retry_buffer {byte7,byte6,byte5,byte4} when a retry_buffer CPU access read/write takes place.

Table 1-260 LPU Txlink Memory Data Load1 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
MEM_RD_WR_DATA1	31:0	rst_1	0x0	RW	Memory Read/Write Data1

1.3.4.162 LPU Txlink Memory Data Load2 Register (0x006E2498, 0x007E2498 / 0x0)

This register is not implemented.

This register stores the read/write data from retry_buffer {byte11,byte10,byte9,byte8} when a retry_buffer CPU access read/write takes place.

Table 1-261 LPU Txlink Memory Data Load2 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:0				Reserved field
None					

1.3.4.163 LPU Txlink Memory Data Load3 Register (0x006E24A0, 0x007E24A0 / 0x0)

This register is not implemented.

This register stores the read/write data from retry_buffer {byte15,byte14,byte13,byte12} when a retry_buffer CPU access read/write takes place.

Table 1-262 LPU Txlink Memory Data Load3 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:0				Reserved field
None					

1.3.4.164 LPU Txlink Memory Data Load4 Register (0x006E24A8, 0x007E24A8 / 0x0)

This is a read only with write for test purpose register. This register stores the read data from retry_buffer parity bits[0:7] when a retry_buffer CPU access read takes place. For test purpose only, this register can also be used to insert parity error in retry buffer location by doing a CPU access write to retry buffer. The parity gets inverted for the bit position which are set during the Rtry Buffer Memory write.

Note: Memory Data Load2 and Load3 Registers are not implemented.

Table 1-263 LPU Txlink Memory Data Load4 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:8				Reserved field
MEM_RD_WR_DATA4	7:0	rst_l	0x0	RW	Memory Read Parity Data4

1.3.4.165 LPU Txlink Retry Data Count Register (0x006E24C0, 0x007E24C0 / 0x0)

This register stores the data bytes in the retry buffer and write for test purposes only.

Table 1-264 LPU Txlink Retry Data Count Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:16				Reserved field
RTRY_DATA_CNT	15:0	rst_l	0x0	RW	Retry Data Count This field is only writable if the retry_disable bit in the pcie_lpu_ll_config register is 1.

1.3.4.166 LPU Txlink Sequence Buffer Count Register (0x006E24C8, 0x007E24C8 / 0x0)

This register stores the count of valid entries in the sequence buffer and write for test purposes only.

Table 1-265 LPU Txlink Sequence Buffer Count Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:12				Reserved field
SEQ_BUFF_CNT	11:0	rst_l	0x0	RW	Sequence Buffer Count Writes only take effect if the retry_disable bit in the pcie_lpu_ll_config register is set.

1.3.4.167 LPU Txlink Sequence Buffer Bottom Data Register (0x006E24D0, 0x007E24D0 / 0x0)

This register stores the bottom data in the sequence buffer and write for test purposes only. Writes to this register only take effect if the retry_disable bit in the pcie_lpu_ll_config register is set.

Table 1-266 LPU Txlink Sequence Buffer Bottom Data Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:31				Reserved field
SBUF_BDATA_PAR	30	rst_l	0x0	RW	Sequence Buffer Bottom Data Parity
SBDATA_SEQ_NUM	29:18	rst_l	0x0	RW	Sequence Buffer Bottom Data Sequence Number

Table 1-266 LPU Txlink Sequence Buffer Bottom Data Register

Field	Bits	Reset Name	Reset Value	Type	Description
SBDATA_RTRY_PTR	17:2	rst_l	0x0	RW	Sequence Buffer Bottom Data Retry Pointer
SBDATA_EOP_POS	1:0	rst_l	0x0	RW	Sequence Buffer Bottom Data EOP Position

1.3.4.168 LPU Txlink Ack Latency Timer Threshold Register (0x006E24E0, 0x007E24E0 / 0x5)

This provides the threshold value in symbol times after which high priority Acks are scheduled when link layer receives good TLP. In order that Acks can be sent out within the Ack latency transmission limit (refer to PCI-Express spec Table3-5), it is recommended to program a low value in this register (not lower than 0x4). The Ack_Latency threshold can be programmed to a higher value for lower link widths and higher max payload size to reduce frequency of sending high priority Acks. This register is not affected by auto-update mechanism.

Table 1-267 LPU Txlink Ack Latency Timer Threshold Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:16				Reserved field
ACK_LAT_THHOLD	15:0	rst_l	0x5	RW	Ack Latency Threshold

1.3.4.169 LPU Rxlink Next Receive Sequence + 1 Counter Register (0x006E2500, 0x007E2500 / 0x1)

For debug and lab use only.

Table 1-268 LPU Rxlink Next Receive Sequence + 1 Counter Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:12				Reserved field
NXT_RX_SEQ_CNTR	11:0	rst_l	0x1	R	Next Receive Sequence Counter This field is 1 greater than the expected Sequence Number for next received TLP.

1.3.4.170 LPU Rxlink Unsupported DLLP Received Register (0x006E2508, 0x007E2508 / 0x0)

This is a Error log register.

Table 1-269 LPU Rxlink Unsupported DLLP Received Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
BYTE3	31:24	rst_1	0x0	R	Byte3
BYTE2	23:16	rst_1	0x0	R	Byte2
BYTE1	15:8	rst_1	0x0	R	Byte1
BYTE0	7:0	rst_1	0x0	R	DLLP received Byte0

1.3.4.171 LPU Rxlink Test Control Register (0x006E2510, 0x007E2510 / 0x0)

Indicated actions performed continuously until cleared by software.

Table 1-270 LPU Rxlink Test Control Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:2				Reserved field
FORCE_SEND_INIT_FC_DLLP	1	rst_1	0x0	RW	Force Send Init Flow Control DLLP info to Trans Layer
FORCE_PAR_ERR_DLLP	0	rst_1	0x0	RW	Force Parity Error for DLLPs

1.3.4.172 LPU Physical Layer Configuration Register (0x006E2600, 0x007E2600 / 0x10)

This register is reset by LPU Reset.rstltssm [4]. See Reset Register.

Table 1-271 LPU Physical Layer Configuration Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field

Table 1-271 LPU Physical Layer Configuration Register

Field	Bits	Reset Name	Reset Value	Type	Description
PHY_TST_EN	31	rst_1	0x0	RW	Physical Test Mode Enable This bit is a mux select for the pcie_lpu_tx_phy_sts_2.recv_det_s ts and pcie_lpu_tx_phy_sts_2.recv_det_raw_sts fields. See the pcie_lpu_tx_phy_sts_2 register for more details.
FAST_SIM	30	rst_1	0x0	RW	Fast Simulation Mode when 1, many register values are forced to shorter values for faster simulation times. If asserted after the SERDES Glue receiver reset is negated, the faster bit and byte sync times will not be used To make the initialization simulation faster: - Assert and negate Reset Receive Lane[15:0] with Global Reset OxpCs to use the new values; - load LTSSM Configuration 2 with 0x0c0; - load LTSSM Configuration 3 with 0x200.
UNUSED	29	rst_1	0x0	RW	Unused Note: never write 1 to this field as this may result in undesired results.
FRCE_EXTEN_SYNC	28	rst_1	0x0	RW	Force Extended Sync Bit when 1, forces the pu_ext_sync signal asserted. when 0, controlled by t2l_link_control[7]. If either is asserted the N_FTS timeout is set to twice 4096 FTS ordered sets. When both are negated, the N_FTS timeout is set to twice the N_FTS value.
RESERVED	27:12				Reserved field

Table 1-271 LPU Physical Layer Configuration Register

Field	Bits	Reset Name	Reset Value	Type	Description
TX_IDLE_POST_EN	11	rst_1	0x0	RW	TX Electrical Idle Postable Enable When 1, a postable of zero's is sent after the electrical idle ordered-set to adjust for electrical idle control requirements.
TX_OS_POST_VAL	10:8	rst_1	0x0	RW	TX Ordered-Set Postable Value Selects the ordered set preamble count when exiting the electrical idle state (IDLE) 0x0 = 1 byte 0x1 = 2 bytes 0x2 = 3 bytes 0x3 = 4 bytes.
TX_OS_BYTE_SEL	7	rst_1	0x0	RW	TX Ordered-Set Byte Select When 1, all training sequences and single lane ordered sets (IDLE_OS & FTS_OS) have the encoded K28.5 10-bit symbol in the lower byte (txdata[10:19]. When 0, the encoded K28.5 10-bit symbol is in the upper byte (txdata[0:9]) which is sent before the lower byte. This may be required to adjust for electrical idle control requirements.
TX_OS_PREAM_VAL	6:4	rst_1	0x1	RW	TX Ordered-Set Preamble Value Selects the ordered set preamble count when exiting the electrical idle state (IDLE). 0x0 = 1 byte 0x1 = 2 bytes 0x2 = 3 bytes 0x3 = 4 bytes.
TX_RDET_BYP_MODE	3	rst_1	0x0	RW	TX RDET Bypass Mode When 1, force the Receiver Detect value to be the max. width select value.

Table 1-271 LPU Physical Layer Configuration Register

Field	Bits	Reset Name	Reset Value	Type	Description
TX_RDET_SAFE_MODE	2	rst_l	0x0	RW	TX RDET Safe Mode This bit is NOT used in current implementation.
TX_UNUSED	1	rst_l	0x0	RW	Unused
TX_PAR_ERR	0	rst_l	0x0	RW	TX Parity Error Replace When 1, a transmit parity error character is replaced with EDB. (K30.7).

1.3.4.173 LPU Phy Layer Status Register (0x006E2608, 0x007E2608 / 0x0)

Reserved for future

Table 1-272 LPU Phy Layer Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:0				Reserved field
None					

1.3.4.174 LPU Phy Layer Interrupt and Status Register (0x006E2610, 0x007E2610 / 0x0)

This register is similar to Sun's error status clear register except Bit [31], which is similar to Sun's interrupt status register.

Note: Bit [8] Training Error maps to l2t_ue_status[0]. All the other bits, [11-9, 7-0] map to l2t_ce_status[0]. This register is reset by LPU Reset.rsterror [8]. See Reset Register.

Table 1-273 LPU Phy Layer Interrupt and Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
INT_PHY_LAYER_ERR	31	por_l	0x0	R	Physical Layer Error Interrupt It's 1 when at least one of the unmasked status bits are asserted and the corresponding bit is unmasked in interrupt mask register.
RESERVED	30:12				Reserved field

Table 1-273 LPU Phy Layer Interrupt and Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
INT_KCHAR_DLLP_ERR	11	por_1	0x0	RW1C	KChar in DLLP Error Note: Don't use it because it's frequently caused by LPU itself.
INT_ILL_END_POS_ERR	10	por_1	0x0	RW1C	Illegal END position Error
INT_LNK_ERR	9	por_1	0x0	RW1C	Link Error When 1, lost byte sync or bit sync or alignment.
INT_TRN_ERR	8	por_1	0x0	RW1C	Training Error
INT_EDB_DET	7	por_1	0x0	RW1C	EDB Detected
INT_SDP_END	6	por_1	0x0	RW1C	SDP without END Note: Don't use it because it's frequently caused by LPU itself.
INT_STP_END_EDB	5	por_1	0x0	RW1C	STP without END or EDB Note: Don't use it because it's frequently caused by LPU itself.
INT_INVLD_CHAR_ERR	4	por_1	0x0	RW1C	Invalid Character Error
INT_MULTI_SDP	3	por_1	0x0	RW1C	Multiple SDPs
INT_MULTI_STP	2	por_1	0x0	RW1C	Multiple STPs
INT_ILL_SDP_POS	1	por_1	0x0	RW1C	Illegal SDP Position
INT_ILL_STP_POS	0	por_1	0x0	RW1C	Illegal STP Position

1.3.4.175 LPU Phy Interrupt and Status Test Register (0x006E2618, 0x007E2618 / 0x0)

This register is similar to Sun's error status set register. This is an error simulation register for the LPU Phy Layer Error Interrupt Register. Writing 1 to each bit sets corresponding bit in the above error register. Reads of this register return 0

Table 1-274 LPU Phy Interrupt and Status Test Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:12				Reserved field
TST_KCHAR_DLLP_ERR	11	rst_1	0x0	RW1S	KChar in DLLP Error

Table 1-274 LPU Phy Interrupt and Status Test Register

Field	Bits	Reset Name	Reset Value	Type	Description
TST_ILL_END_POS_ERR	10	rst_l	0x0	RW1S	Illegal END position Error
TST_LNK_ERR	9	rst_l	0x0	RW1S	Link Error
TST_TRN_ERR	8	rst_l	0x0	RW1S	Training Error
TST_EDB_DET	7	rst_l	0x0	RW1S	EDB Detected
TST_SDP_END	6	rst_l	0x0	RW1S	SDP without END
TST_STP_END_EDB	5	rst_l	0x0	RW1S	STP without END or EDB
TST_INVLD_CHAR_ERR	4	rst_l	0x0	RW1S	Invalid Character Error
TST_MULTI_SDP	3	rst_l	0x0	RW1S	Multiple SDPs
TST_MULTI_STP	2	rst_l	0x0	RW1S	Multiple STPs
TST_ILL_SDP_POS	1	rst_l	0x0	RW1S	Illegal SDP Position
TST_ILL_STP_POS	0	rst_l	0x0	RW1S	Illegal STP Position

1.3.4.176 LPU Phy Interrupt Mask Register (0x006E2620, 0x007E2620 / 0x80000FFF)

This register is similar to Sun's interrupt enable register. Write 0 to enable interrupt and 1 to disable interrupt. This register is reset by LPU Reset.rstltssm [4]. See Reset Register.

Table 1-275 LPU Phy Interrupt Mask Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
MSK_PHY_LAYER_ERR	31	rst_l	0x1	RW	Physical Layer Error
RESERVED	30:12				Reserved field
MSK_KCHAR_DLLP_ERR	11	rst_l	0x1	RW	KChar in DLLP Error
MSK_ILL_END_POS_ERR	10	rst_l	0x1	RW	Illegal END position Error
MSK_LNK_ERR	9	rst_l	0x1	RW	Link Error
MSK_TRN_ERR	8	rst_l	0x1	RW	Training Error
MSK_EDB_DET	7	rst_l	0x1	RW	EDB Detected

Table 1-275 LPU Phy Interrupt Mask Register

Field	Bits	Reset Name	Reset Value	Type	Description
MSK_SDP_END	6	rst_l	0x1	RW	SDP without END
MSK_STP_END_EDB	5	rst_l	0x1	RW	STP without END or EDB
MSK_INVLD_CHAR_ERR	4	rst_l	0x1	RW	Invalid Character Error
MSK_MULTI_SDP	3	rst_l	0x1	RW	Multiple SDPs
MSK_MULTI_STP	2	rst_l	0x1	RW	Multiple STPs
MSK_ILL_SDP_POS	1	rst_l	0x1	RW	Illegal SDP Position
MSK_ILL_STP_POS	0	rst_l	0x1	RW	Illegal STP Position

1.3.4.177 LPU Receive Phy Config Register (0x006E2680, 0x007E2680 / 0x0)

This register is for debug and lab purpose with the exception of the Water Mark Select which may be for static config. This register is reset by LPU Reset.rstltssm [4]. See Reset Register.

Table 1-276 LPU Receive Phy Config Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
RX_PHY_TST	31	rst_l	0x0	RW	Receive Physical Test Mode
UNUSED_0	30:18	rst_l	0x0	RW	Unused
WM_SEL_FIFO	17:16	rst_l	0x0	RW	Water Mark Select For Elasticity Fifo 2'b00 - 5 2'b01 - 6 2'b10 - 7 2'b11 - 8
UNUSED_1	15:8	rst_l	0x0	RW	Unused, it used to be to reset Receive Lane(x) in x16 LPU core

Table 1-276 LPU Receive Phy Config Register

Field	Bits	Reset Name	Reset Value	Type	Description
RST_RCV_LANE	7:0	rst_l	0x0	RW	<p>Reset Receive Lane(x)</p> <p>When 1, the SERDES receiver reset is asserted and the SERDES Glue receive logic for the lane is reset. These bits are also reset when ur_rxpcs_rst (REG_reset[0]) is asserted. The SERDES receive analog reset is only asserted by signal t2l_por.</p> <p>[07] Reset Receive Lane_7 RXPHY_CONF_RCV_RESET_LANE7</p> <p>[06] Reset Receive Lane_6 RXPHY_CONF_RCV_RESET_LANE6</p> <p>[05] Reset Receive Lane_5 RXPHY_CONF_RCV_RESET_LANE5</p> <p>[04] Reset Receive Lane_4 RXPHY_CONF_RCV_RESET_LANE4</p> <p>[03] Reset Receive Lane_3 RXPHY_CONF_RCV_RESET_LANE3</p> <p>[02] Reset Receive Lane_2 RXPHY_CONF_RCV_RESET_LANE2</p> <p>[01] Reset Receive Lane_1 RXPHY_CONF_RCV_RESET_LANE1</p> <p>[00] Reset Receive Lane_0 RXPHY_CONF_RCV_RESET_LANE0</p> <p>Note for simulation: in fast sim mode, reset receive lanes after setting new value in LPU LTSSM Config2 Register.</p>

1.3.4.178 LPU Receive Phy Status1 Register (0x006E2688, 0x007E2688 / 0x0)

Write to this register is for test purpose only.

Table 1-277 LPU Receive Phy Status1 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:17				Reserved field
ALIGN_STS	16	rst_1	0x0	R	Align Status Asserted when all enabled lanes are aligned.
RX_PHY_STS	15:0	rst_1	0x0	RW	Receive Phy Status to LTSSM This status is connected to the LTSSM interrupt register and are defined there. Most of these signals are pulsed and can not be read by the processor.

1.3.4.179 LPU Receive Phy Status2 Register (0x006E2690, 0x007E2690 / 0x0)

This register contains the Training Sequence fields received by this Port lane_0. See LTSSM Config4 register for the values transmitted.

Table 1-278 LPU Receive Phy Status2 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:28				Reserved field
RCV_DIS_SCRAM	27	rst_1	0x0	R	Received Disable Scramble Disable scrambling from TSx Ordered Set on Master Lane. Applies to Rx and Tx PCS. Overridden in 10-bit parallel Rx to Tx loopback mode.
RCV_EN_LOOPBACK	26	rst_1	0x0	R	Received Enable Loopback Enable loopback from TSx Ordered Set on Master Lane.
RCV_DIS_LINK	25	rst_1	0x0	R	Received Disable Link Disable Link from TSx Ordered Set on Master Lane.

Table 1-278 LPU Receive Phy Status2 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RCV_HOT_RST	24	rst_l	0x0	R	Received Hot Reset Reset from TSx Ordered Set on master Lane.
RCV_DATA_RATE	23:16	rst_l	0x0	R	Received Data Rate Date Rate from TSx Ordered Set on Master Lane.
RCV_FTS_NUM	15:8	rst_l	0x0	R	Received FTS Number N_FTS from TSx Ordered Set on master lane
RCV_LINK_NUM	7:0	rst_l	0x0	R	Received Link Number Link Number from TSx Ordered set on Master Lane.

1.3.4.180 LPU Receive Phy Status3 Register (0x006E2698, 0x007E2698 / 0x0)

This is LPU Receive Phy Status3 Register.

Table 1-279 LPU Receive Phy Status3 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:24				Reserved field
POL_REV_STS	23:16	rst_l	0x0	R	Polarity Reversal Status Lanes 7 to 0
RESERVED	15:8				Reserved field
BYTE_SYNC_STS	7:0	rst_l	0x0	R	Byte Synchronization Status Lanes 7 to 0

1.3.4.181 LPU Receive Phy Interrupt and Status Register (0x006E26A0, 0x007E26A0 / 0x0)

This register is similar to Sun's error status clear register except Bit [31], which is similar to Sun's interrupt status register. This register is reset by LPU Reset.rsterror [8]. See Reset Register.

Table 1-280 LPU Receive Phy Interrupt and Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
INT_RCV_PHY	31	por_1	0x0	R	Receive Phy Interrupt It's 1 when at least one of the unmasked status bits are asserted and the corresponding bit is unmasked in interrupt mask register.
RESERVED	30:12				Reserved field
INT_UNUSED	11:3	por_1	0x0	RW1C	Unused
INT_ALIGN_ERR	2	por_1	0x0	RW1C	Alignment Error
INT_ELSTC_FIFO_OVRFLW	1	por_1	0x0	RW1C	Elastic FIFO Overflow
INT_ELSTC_FIFO_UNDRFLW	0	por_1	0x0	RW1C	Elastic FIFO Underflow

1.3.4.182 LPU Receive Phy Interrupt and Status Test Register (0x006E26A8, 0x007E26A8 / 0x0)

This register is similar to Sun's error status set register. Reads of this register return 0.

Table 1-281 LPU Receive Phy Interrupt and Status Test Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:12				Reserved field
TST_UNUSED	11:3	rst_1	0x0	RW1S	Unused
TST_ALIGN_ERR	2	rst_1	0x0	RW1S	Alignment Error
TST_ELSTC_FIFO_OVRFLW	1	rst_1	0x0	RW1S	Elastic FIFO Overflow
TST_ELSTC_FIFO_UNDRFLW	0	rst_1	0x0	RW1S	Elastic FIFO Underflow

1.3.4.183 LPU Receive Phy Interrupt Mask Register (0x006E26B0, 0x007E26B0 / 0x80000FFF)

This register is similar to Sun's interrupt enable register. Write 0 to enable interrupt and 1 to disable interrupt. This register is reset by LPU Reset.rstltssm [4]. See Reset Register.

Table 1-282 LPU Receive Phy Interrupt Mask Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
MSK_RCV_PHY_INT	31	rst_l	0x1	RW	Receive Phy Interrupt Mask
RESERVED	30:12				Reserved field
MSK_UNUSED	11:3	rst_l	0x1FF	RW	Unused These bits correspond to unimplemented interrupt events.
MSK_ALIGN_ERR	2	rst_l	0x1	RW	Alignment Error Mask
MSK_ELSTC_FIFO_OVRFLW	1	rst_l	0x1	RW	Elastic FIFO Overflow Mask
MSK_ELSTC_FIFO_UNDRFLW	0	rst_l	0x1	RW	Elastic FIFO Underflow Mask

1.3.4.184 LPU Transmit Phy Config Register (0x006E2700, 0x007E2700 / 0x0)

This register is reset by LPU Reset.rstltssm [4]. See Reset Register.

Table 1-283 LPU Transmit Phy Config Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
FRCE_RCVR_DET	31:16	rst_l	0x0	RW	Force Receiver Detect when 1, receiver detect and electrical idle are asserted for that lane. The receiver detect status is updated after the SERDES receiver detect timeout delay. This control overrides TXPHY_CONF_TX_PCS_ELEC_IDLE. See SERDES glue power down 2 register.

Table 1-283 LPU Transmit Phy Config Register

Field	Bits	Reset Name	Reset Value	Type	Description
FRCE_ELEC_IDLE	15:0	rst_l	0x0	RW	Force Electrical Idle when 1, electrical idle is asserted for that lane. This control overrides other controls except TXPHY_CONF_TX_PCS_RCVR_DET.

1.3.4.185 LPU Transmit Phy Status Register (0x006E2708, 0x007E2708 / 0x73000010)

This register is reset by LPU Reset.rsttxphy [3]. See Reset Register.

Table 1-284 LPU Transmit Phy Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
NEG_LANE_WIDTH	31:28	rst_l	0x7	R	Negotiated Lane Width The default value is specific for x8 core.
TXPHY_SCRAM_EN	27	rst_l	0x0	R	TXPHY Scramble Enable When 1, the transmit data will be scrambled.
TX_LANE_REV	26	rst_l	0x0	R	Transmit Lane Reversal
TX_LANE_PAD	25	rst_l	0x1	R	Transmit TS1/TS2 Lane as PAD
TX_LINK_PAD	24	rst_l	0x1	R	Transmit TS1/TS2 Link as PAD
RESERVED	23				Reserved field
TX_PHY_SMS	22:0	rst_l	0x10	R	Transmit Phy State machine State This is a one-hot state machine.

1.3.4.186 LPU Transmit Phy Interrupt and Status Register (0x006E2710, 0x007E2710 / 0x0)

Many of these interrupts are asserted under normal condition and all are normally masked. This register is similar to Sun's error status clear register except Bit [31], which is similar to Sun's interrupt status register. This register is reset by LPU Reset.rsterror [8]. See Reset Register.

Table 1-285 LPU Transmit Phy Interrupt and Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
INT_UNMSK	31	por_1	0x0	R	Any Unmasked Interrupt when 1, one of the unmasked interrupt is active
RESERVED	30:12				Reserved field
INT_RCV_IDLE	11	por_1	0x0	RW1C	Received IDLE Interrupt
INT_RCV_TS2	10	por_1	0x0	RW1C	Received TS2 Interrupt
INT_RCV_TS1	9	por_1	0x0	RW1C	Received TS1 Interrupt
INT_SKP_ERR	8	por_1	0x0	RW1C	Skip Error Interrupt when 1, the number of outstanding skips is 6 or greater.
INT_SKP_DONE_BK2BK	7	por_1	0x0	RW1C	Skip Done Back to Back set when two back to back skips have been sent.
INT_SKP_ACK_DECR	6	por_1	0x0	RW1C	Skip ACK Decrement set when the Link allows a skip to be inserted.
INT_SKP_DONE_DECR	5	por_1	0x0	RW1C	Skip Done Decrement set when the last Skip of the ordered set is sent.
INT_SKP_TRIG	4	por_1	0x0	RW1C	Skip Trigger set when it is time to send a Skip ordered set. set every 1180 characters on a single lane (325 characters if pu_sim_fast is 1).
INT_UNUSED_2	3:2	por_1	0x0	RW1C	Unused Interrupt 2

Table 1-285 LPU Transmit Phy Interrupt and Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
INT_RCVR_DET_VALID	1	por_1	1'bx	RW1C	Receiver Detect Value is Valid Interrupt set when the receiver detect process has completed. Note: reset value as x because the duration to complete the receiver detect process is unknown. i.e. transition from 0 to 1.
INT_TX_PAR_ERR	0	por_1	0x0	RW1C	Transmit Parity Error Interrupt set when a transmit parity error is detected on any lane.

1.3.4.187 LPU Transmit Phy Interrupt and Status Test Register (0x006E2718, 0x007E2718 / 0x0)

This is an error simulation register for the LPU Transmit Phy Interrupt Register. Writing 1 to each bit sets corresponding bit in the above error register. It is similar to Sun's error status set register. Reads of this register return 0.

Table 1-286 LPU Transmit Phy Interrupt and Status Test Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:12				Reserved field
TST_TX_PHY_INT	11:0	rst_1	0x0	RW1S	Transmit Phy Interrupt Test Bits Writing 1 to each bit sets corresponding bit in Transmit Phy Interrupt Register.

1.3.4.188 LPU Transmit Phy Interrupt Mask Register (0x006E2720, 0x007E2720 / 0x80000FFF)

This register is similar to Sun's interrupt enable register. Write 0 to enable interrupt and 1 to disable interrupt. This register is reset by LPU Reset.rstltssm [4]. See Reset Register.

Table 1-287 LPU Transmit Phy Interrupt Mask Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
MSK_GLOBL_INT	31	rst_1	0x1	RW	Global Interrupt Mask

Table 1-287 LPU Transmit Phy Interrupt Mask Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	30:12				Reserved field
MSK_IMPLM_INT	11:0	rst_l	0xFFFF	RW	Interrupt Mask Bits Setting any of these bits masks interrupts from corresponding bits in Transmit Phy Interrupt and status Register.

1.3.4.189 LPU Transmit Phy Status 2 Register (0x006E2728, 0x007E2728 / 0x0)

This register contains the receiver detect raw and registered status, based on phy_tst_en field of the pcie_lpu_phy_cnfg register. This register is reset by LPU Reset.rsttxphy [3]. See Reset Register.

Table 1-288 LPU Transmit Phy Status 2 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
RECV_DET_STS	31:16	rst_l	16'bx	R	When pcie_lpu_phy_cnfg.phy_tst_en is 1, this field reflects the state of the s2p_rxpres[15:0] signals from the serdes. When pcie_lpu_phy_cnfg.phy_tst_en is 0, this field reflects the state of the txphy.tx_ctrl.pt_rcvdet_maskp[15:0] signals.
RECV_DET_RAW_STS	15:0	rst_l	16'bx	R	When pcie_lpu_phy_cnfg.phy_tst_en is 1, this field reflects the state of the s2p_rxpresvalid[15:0] signals from the serdes. When pcie_lpu_phy_cnfg.phy_tst_en is 0, this field reflects the state of the txphy.tx_ctrl.pt_rcvterm_enablep[15:0] signals.

1.3.4.190 LPU LTSSM Config1 Register (0x006E2780, 0x007E2780 / 0x1905)

This register is for debug and lab use only. This register is reset by LPU Reset.rstltssm [4]. See Reset Register.

Table 1-289 LPU LTSSM Config1 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
LTSSM_TST	31	rst_l	0x0	RW	LTSSM Test Mode This bit, when asserted, enables writes to the LTSSM status register fields.
CFG_UNUSED	30:18	rst_l	0x0	RW	Unused
LPBK_MSTR	17	rst_l	0x0	RW	Loopback Master This bit should be written as 0 in normal functional mode. Setting this to 1 is not supported in Fire and will result in undefined behaviour.
HI_DATA_SUP	16	rst_l	0x0	RW	Higher Data Rate Supported This bit should be written as 0 in normal functional mode. Setting this to 1 is not supported in Fire and will result in undefined behaviour. This bit is used for LTSSM state machine debug control only.
LTSSM_8_TO	15:8	rst_l	0x19	RW	LTSSM 8 ns Timeout value This timeout value is used within the LTSSM to wait 8 nsec where specified (8 ns + a maximum of 50%). This value must be greater than the rxlos filter delay. This value is the PCIE spec Ttx-idle-setto-min value for 20-bits or 8 nsec delay. The maximum value is 1.024 usec at 250 MHz.

Table 1-289 LPU LTSSM Config1 Register

Field	Bits	Reset Name	Reset Value	Type	Description
LTSSM_20_TO	7:0	rst_l	0x5	RW	LTSSM 20 ns Timeout value This timeout value is used within the LTSSM to wait 20 nsec where specified (20 ns + a maximum of 50%). This value is the PCIE spec Ttxidleaset-to-min value for 50-bit or 20 nsec delay. This is the LTSSM RXE to RXI timeout delay before waiting to check for RXLOS=0.

1.3.4.191 LPU LTSSM Config2 Register (0x006E2788, 0x007E2788 / 0x2DC6C0)

This register is reset by LPU Reset.rstltssm [4]. See Reset Register.

Table 1-290 LPU LTSSM Config2 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
LTSSM_12_TO	31:0	rst_l	0x2DC6C0	RW	LTSSM 12 ms Timeout value (12 ms + a maximum of 50%) This value determines how long the LTSSM waits before sampling signals during initialization. A multiple of this value is used for the 24 ms and 48 ms timeout values. Set this value to 0x200 for most simulations, depending on how fast the other port takes for each state. A value of 0x200 works well in pad or ewrap loopback modes. Note for simulations only: in core, lpu-lpu and chip environments, 0x400 works well.

1.3.4.192 LPU LTSSM Config3 Register (0x006E2790, 0x007E2790 / 0x7A120)

This register is reset by LPU Reset.rstltssm [4]. See Reset Register.

Table 1-291 LPU LTSSM Config3 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
LTSSM_2_TO	31:0	rst_l	0x7A120	RW	LTSSM 2 ms Timeout value (2 ms + a maximum of 50%). This value is used to wait for valid received link and port numbers during configuration. Set this value to 0x300 or more for faster simulations, depending on how quick valid numbers are provided in the test. A value of 0x300 works well in pad or ewrap loopback modes.

1.3.4.193 LPU LTSSM Config4 Register (0x006E2798, 0x007E2798 / 0x29C00)

The content of this register determines what is placed in corresponding field of the TS1/TS2 Ordered Set transmitted. See also the LPU Receive Phy Status2 Register. This register is reset by LPU Reset.rstltssm [4]. See Reset Register.

Table 1-292 LPU LTSSM Config4 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
TRN_CNTRL	31:24	rst_l	0x0	RW	Transmit Training Control This is the value transmitted in the Training Sequence TS1/TS2 Control field by this Port. See receive phy status 2 register. Note: don't write to this field.
DATA_RATE	23:16	rst_l	0x2	RW	Transmit Data Rate This is the value transmitted in the Training Sequence TS1/TS2 Data Rate field by this Port.

Table 1-292 LPU LTSSM Config4 Register

Field	Bits	Reset Name	Reset Value	Type	Description
N_FTS	15:8	rst_l	0x9C	RW	Transmit FTS Number This is the N_FTS value which is transmitted in the TS1/TS2 ordered sets by this port. This value is dependent on the bit lock time in SERDES Glue Configuration register. The default value is 0x9c for a 5000-bit lock time. Use 0x4c for a 2500-bit lock time. This value is equal to the number of bits to lock to the receive data + 50-bit times divided by 40-bits per FTS ordered set.
LNK_NUM	7:0	rst_l	0x0	RW	Link Number This is the Link Number transmitted by this port during the TS1/TS2 ordered set.

1.3.4.194 LPU LTSSM Config5 Register (0x006E27A0, 0x007E27A0 / 0x0)

This register is for State Transition Control. This register is reset by LPU Reset.rstltssm [4]. See Reset Register.

Note: In normal operation, set bit[11] to 1'b1 to disable LTSSM entering Polling.Compliance state since Polling.Compliance is intended for a compliance test environment. The argument for disabling entry into Polling.Compliance is that the PCI-E spec indicates that if ANY lane detects a receiver and only receives electric-idle during Polling.Active, then the entire link falls into Polling.Compliance and stays there until soft reset. This 'rule' introduces eight single points of failure (i.e. each of the remote device's transmitters). Ignoring the rule (by disabling entry to Polling.Compliance) allows the link to configure without these no-TX lanes.

Table 1-293 LPU LTSSM Config5 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
CFG_UNUSED_0	31:13	rst_l	0x0	RW	Unused

Table 1-293 LPU LTSSM Config5 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RCV_DET_TST_MODE	12	rst_1	0x0	RW	Receiver Detect Test Mode When 1, the second receiver detect request is bypassed for all maximum link widths.
POLL_CMPLNC_DIS	11	rst_1	0x0	RW	Polling Compliance Disable When 1, the Compliance Pattern State entry is disabled. Suggested set to 1.
TX_IDLE_TX_FTS	10	rst_1	0x0	RW	Tx_L0s.Idle to Tx_L0s.FTS
RX_FTS_RVR_LK	9	rst_1	0x0	RW	Rx_L0s.FTS to Recovery.Rcvr-Lock
CFG_UNUSED_1	8:7	rst_1	0x0	RW	Unused
LPBK_ENTRY_ACTIVE	6	rst_1	0x0	RW	Loopback.Entry to Loopback.Active
LPBK_ENTRY_EXIT	5	rst_1	0x0	RW	Loopback.Entry to Loopback.Exit
LPBK_ACTIVE_EXIT	4	rst_1	0x0	RW	Loopback.Active to Loopback.Exit
L1_IDLE_RCVRY_LK	3	rst_1	0x0	RW	L1.Idle to Recovery.RcvrLock
L0_TRN_CNTRL_RST	2	rst_1	0x0	RW	L0 to Training Control Reset
L0_LPBK	1	rst_1	0x0	RW	L0 to Loopback
CFG_UNUSED_2	0	rst_1	0x0	RW	Unused

1.3.4.195 LPU LTSSM Status1 Register (0x006E27A8, 0x007E27A8 / 0x0)

Write to this register is for test purpose only. For writes to this register to take effect, both the LTSSM_TST (bit 31) field of the LPU LTSSM Config1 Register as well as the appropriate fields in the LPU LTSSM Status Write Enable Register must first be set.

Table 1-294 LPU LTSSM Status1 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field

Table 1-294 LPU LTSSM Status1 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RX_LN_EN_MSK	31:16	rst_1	0x0	RW	Receive Lane Enable Mask These signals are used by the receive PCS for lane alignment. It is a product of the lanes enabled during receiver detect and the maximum lanes supported.
RX_ALGN_CMD	15	rst_1	0x0	RW	Receive Align Command
MSTR_LN_SEL	14	rst_1	0x0	RW	Master Lane Select When 0, lane_0 is the master lane which provides most of the configuration information. When 1, the maximum lane is the master.
LNK_OT_RX	13	rst_1	0x0	RW	Link Open To Receive
LNK_OT_TX	12	rst_1	0x0	RW	Link Open To Transmit
LN_RVRSD	11	rst_1	0x0	RW	Lane Reversed Note: this bit is not connected in HW. it's here for future use.
LNK_UP_DWN_STS	10	rst_1	0x0	RW	Link Up Down Status

Table 1-294 LPU LTSSM Status1 Register

Field	Bits	Reset Name	Reset Value	Type	Description
LTSSM_STATE	9:4	rst_l	0x0	RW	LTSSM State 6'h00 DETECT_QUIET 6'h01 DETECT_ACTIVE 6'h02 DETECT_CHARGE 6'h03 POLLING_QUIET 6'h04 POLLING_ACTIVE 6'h05 POLLING_COMPL 6'h06 POLLING_CONFIG 6'h07 POLLING_SPEED 6'h08 CONFIG_RCVRCFG 6'h09 CONFIG_IDLE 6'h0a RECOVERY_RCVRLOCK 6'h0b RECOVERY_RCVRCFG 6'h0c RECOVERY_IDLE 6'h0d L0 6'h0e L0S_RXE 6'h0f L0S_RXI 6'h10 L0S_RXF 6'h11 L0S_TXE 6'h12 L0S_TXI 6'h13 L0S_TXF 6'h14 L1_ENTRY 6'h15 L1_IDLE 6'h16 Reserved 6'h17 Reserved 6'h18 Reserved 6'h19 DISABLED 6'h1a LPBACK_ENTRY 6'h1b LPBACK_ACTIVE 6'h1c LPBACK_EXIT 6'h1d TCRESET 6'h1e L0S_RXE_TXE 6'h1f L0S_RXE_TXI 6'h20 L0S_RXE_TXF 6'h21 L0S_RXI_TXE 6'h22 L0S_RXI_TXI 6'h23 L0S_RXI_TXF 6'h24 L0S_RXF_TXE 6'h25 L0S_RXF_TXI 6'h26 L0S_RXF_TXF 6'h27 - 6'h3f Reserved

Table 1-294 LPU LTSSM Status1 Register

Field	Bits	Reset Name	Reset Value	Type	Description
CNFG_LNK_WIDTH	3:0	rst_l	4'bx	RW	Configured Link Width This indicates the current Link Width size in the form of n-1.

1.3.4.196 LPU LTSSM Status2 Register (0x006E27B0, 0x007E27B0 / 0x0)

Write to this register is for test purpose only. For writes to this register to take effect, both the LTSSM_TST (bit 31) field of the LPU LTSSM Config1 Register as well as the appropriate fields in the LPU LTSSM Status Write Enable Register must first be set.

Table 1-295 LPU LTSSM Status2 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
TX_CMD_TX_PHY	31:16	rst_l	16'bx	RW	Transmit Command to Transmit Phy
RX_CMD_RX_PHY	15:0	rst_l	16'bx	RW	Receive Command to Receive Phy

1.3.4.197 LPU LTSSM Interrupt and Status Register (0x006E27B8, 0x007E27B8 / 0x0)

This register is similar to Sun's error status clear register except Bit [31], which is similar to Sun's interrupt status register. This register is reset by LPU Reset.rsterror [8]. See Reset Register.

Table 1-296 LPU LTSSM Interrupt and Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
INT_ANY	31	por_l	0x0	R	Any LTSSM interrupt active It's 1 when at least one of the unmasked status bits are asserted and the corresponding bit is unmasked in interrupt mask register.
RESERVED	30:16				Reserved field
INT_SKIP_OS	15	por_l	0x0	RW1C	LTSSM_INTR_RCVD_SKIP_OS Received

Table 1-296 LPU LTSSM Interrupt and Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
INT_FTS	14	por_1	0x0	RW1C	LTSSM_INTR_RCVD_FTS Received
INT_TS2_RECOV	13	por_1	0x0	RW1C	LTSSM_INTR_RCVD_TS2_RECOV Received
INT_8IDLE_DATA	12	por_1	0x0	RW1C	LTSSM_INTR_RCVD_8IDLE_DATA Received
INT_IDLE_DATA	11	por_1	0x0	RW1C	LTSSM_INTR_RCVD_IDLE_DATA Receive
INT_TSX_POLL	10	por_1	0x0	RW1C	LTSSM_INTR_RCVD_TSX_POLL Received
INT_TSX_INV	9	por_1	0x0	RW1C	LTSSM_INTR_RCVD_TSX_INV Received
INT_EIDLE_EXIT	8	por_1	0x0	RW1C	LTSSM_INTR_RCVD_EIDLE_EXIT Received
INT_TSX_COMP	7	por_1	0x0	RW1C	LTSSM_INTR_RCVD_TSX_COMP Received
INT_TSX_LB	6	por_1	0x0	RW1C	LTSSM_INTR_RCVD_TSX_LB Received
INT_TSX_DIS	5	por_1	0x0	RW1C	LTSSM_INTR_RCVD_TSX_DIS Received
INT_TSX_RST	4	por_1	0x0	RW1C	LTSSM_INTR_RCVD_TSX_RST Received
INT_EIDLE	3	por_1	0x0	RW1C	LTSSM_INTR_RCVD_EIDLE Received
INT_TS2	2	por_1	0x0	RW1C	LTSSM_INTR_RCVD_TS2 Received
INT_TS1	1	por_1	0x0	RW1C	LTSSM_INTR_RCVD_TS1 Received
INT_NONE	0	por_1	0x0	RW1C	LTSSM_INTR_NONE Received

1.3.4.198 LPU LTSSM Interrupt and Status Test Register (0x006E27C0, 0x007E27C0 / 0x0)

This register is similar to Sun's error status set register. Reads of this register return 0.

Table 1-297 LPU LTSSM Interrupt and Status Test Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:16				Reserved field
TST_SKIP_OS	15	rst_1	0x0	RW1S	LTSSM_INTR_RCVD_SKIP_OS Received
TST_FTS	14	rst_1	0x0	RW1S	LTSSM_INTR_RCVD_FTS Received
TST_TS2_RECOV	13	rst_1	0x0	RW1S	LTSSM_INTR_RCVD_TS2_RECOV Received
TST_8IDLE_DATA	12	rst_1	0x0	RW1S	LTSSM_INTR_RCVD_8IDLE_DATA Received
TST_IDLE_DATA	11	rst_1	0x0	RW1S	LTSSM_INTR_RCVD_IDLE_DATA Receive
TST_TSX_POLL	10	rst_1	0x0	RW1S	LTSSM_INTR_RCVD_TSX_POLL Received
TST_TSX_INV	9	rst_1	0x0	RW1S	LTSSM_INTR_RCVD_TSX_INV Received
TST_EIDLE_EXIT	8	rst_1	0x0	RW1S	LTSSM_INTR_RCVD_EIDLE_EXIT Received
TST_TSX_COMP	7	rst_1	0x0	RW1S	LTSSM_INTR_RCVD_TSX_COMP Received
TST_TSX_LB	6	rst_1	0x0	RW1S	LTSSM_INTR_RCVD_TSX_LB Received
TST_TSX_DIS	5	rst_1	0x0	RW1S	LTSSM_INTR_RCVD_TSX_DIS Received
TST_TSX_RST	4	rst_1	0x0	RW1S	LTSSM_INTR_RCVD_TSX_RST Received
TST_EIDLE	3	rst_1	0x0	RW1S	LTSSM_INTR_RCVD_EIDLE Received
TST_TS2	2	rst_1	0x0	RW1S	LTSSM_INTR_RCVD_TS2 Received

Table 1-297 LPU LTSSM Interrupt and Status Test Register

Field	Bits	Reset Name	Reset Value	Type	Description
TST_TS1	1	rst_l	0x0	RW1S	LTSSM_INTR_RCVD_TS1 Received
TST_NONE	0	rst_l	0x0	RW1S	LTSSM_INTR_NONE Received

1.3.4.199 LPU LTSSM Interrupt Mask Register (0x006E27C8, 0x007E27C8 / 0x8000FFFF)

This register is similar to Sun's interrupt enable register. Write 0 to enable interrupt and 1 to disable interrupt. This register is reset by LPU Reset.rstltssm [4]. See Reset Register.

Table 1-298 LPU LTSSM Interrupt Mask Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
MSK_GLB	31	rst_l	0x1	RW	Global LTSSM interrupt Mask
RESERVED	30:16				Reserved field
MSK_SKIP_OS	15	rst_l	0x1	RW	LTSSM_INTR_RCVD_SKIP_OS Received
MSK_FTS	14	rst_l	0x1	RW	LTSSM_INTR_RCVD_FTS Received
MSK_TS2_RECOV	13	rst_l	0x1	RW	LTSSM_INTR_RCVD_TS2_RECOV Received
MSK_8IDLE_DATA	12	rst_l	0x1	RW	LTSSM_INTR_RCVD_8IDLE_DATA Received
MSK_IDLE_DATA	11	rst_l	0x1	RW	LTSSM_INTR_RCVD_IDLE_DATA Receive
MSK_TSX_POLL	10	rst_l	0x1	RW	LTSSM_INTR_RCVD_TSX_POLL Received
MSK_TSX_INV	9	rst_l	0x1	RW	LTSSM_INTR_RCVD_TSX_INV Received
MSK_EIDLE_EXIT	8	rst_l	0x1	RW	LTSSM_INTR_RCVD_EIDLE_EXIT Received

Table 1-298 LPU LTSSM Interrupt Mask Register

Field	Bits	Reset Name	Reset Value	Type	Description
MSK_TSX_COMP	7	rst_l	0x1	RW	LTSSM_INTR_RCVD_TSX_COMP Received
MSK_TSX_LB	6	rst_l	0x1	RW	LTSSM_INTR_RCVD_TSX_LB Received
MSK_TSX_DIS	5	rst_l	0x1	RW	LTSSM_INTR_RCVD_TSX_DIS Received
MSK_TSX_RST	4	rst_l	0x1	RW	LTSSM_INTR_RCVD_TSX_RST Received
MSK_EIDLE	3	rst_l	0x1	RW	LTSSM_INTR_RCVD_EIDLE Received
MSK_TS2	2	rst_l	0x1	RW	LTSSM_INTR_RCVD_TS2 Received
MSK_TS1	1	rst_l	0x1	RW	LTSSM_INTR_RCVD_TS1 Received
MSK_NONE	0	rst_l	0x1	RW	LTSSM_INTR_NONE Received

1.3.4.200 LPU LTSSM Status Write Enable Register (0x006E27D0, 0x007E27D0 / 0x0)

This register contains the write enables for several Read-Only-Write for test registers. This register is reset by LPU Reset.rstltssm [4]. See Reset Register.

Table 1-299 LPU LTSSM Status Write Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
WE_UNUSED	31:11	rst_l	0x0	RW	Unused
WE1_LTSSM_STS2	10	rst_l	0x0	RW	Write Enable for LTSSM Status2.tx_cmd_tx_phy [31:16]
WE2_LTSSM_STS2	9	rst_l	0x0	RW	Write Enable for LTSSM Status2.rx_cmd_rx_phy [15:0]
WE1_LTSSM_STS1	8	rst_l	0x0	RW	Write Enable for LTSSM Status1.rx_ln_en_msk [31:16]

Table 1-299 LPU LTSSM Status Write Enable Register

Field	Bits	Reset Name	Reset Value	Type	Description
WE2_LTSSM_STS1	7	rst_l	0x0	RW	Write Enable for LTSSM Status1.rx_algn_cmd [15]
WE3_LTSSM_STS1	6	rst_l	0x0	RW	Write Enable for LTSSM Status1.mstr_ln_sel [14]
WE4_LTSSM_STS1	5	rst_l	0x0	RW	Write Enable for LTSSM Status1.lnk_ot_rx [13]
WE5_LTSSM_STS1	4	rst_l	0x0	RW	Write Enable for LTSSM Status1.lnk_ot_tx [12]
WE6_LTSSM_STS1	3	rst_l	0x0	RW	Write Enable for LTSSM Status1.ln_rvrsd [11]
WE7_LTSSM_STS1	2	rst_l	0x0	RW	Write Enable for LTSSM Status1.lnk_up_dwn_sts [10]
WE8_LTSSM_STS1	1	rst_l	0x0	RW	Write Enable for LTSSM Status1.ltssm_state [9:4]
WE9_LTSSM_STS1	0	rst_l	0x0	RW	Write Enable for LTSSM Status1.cnfg_lnk_width [3:0]

1.3.4.201 LPU SERDES Glue Config1 Register (0x006E2800, 0x007E2800 / 0x89019)

This register is reset by LPU Reset.rstltssm [4]. See Reset Register.

Table 1-300 LPU SERDES Glue Config1 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
UNUSED_CNTL1	31:28	rst_l	0x0	RW	Unused Control Bits
STM_SEL	27:24	rst_l	0x0	RW	Self Test Mode Select [0:3]
UNUSED_CNTL2	23:22	rst_l	0x0	RW	Unused Control Bits
REV_LPBK_SEL	21:20	rst_l	0x0	RW	Reverse Loopback Tx Clock Select Selects Transmitter Reference clock in Reverse Loopback mode.

Table 1-300 LPU SERDES Glue Config1 Register

Field	Bits	Reset Name	Reset Value	Type	Description
REV_LPBK_MODE	19	rst_l	0x1	RW	Reverse Loopback Mode 0 - 8 bit reverse PCS loopback, 1 - 10 bit normal loopback.
LPBK_ENB	18	rst_l	0x0	RW	Loopback Enable The LTSSM overrides this value and forces pl_loopback_enablep (rtl signal) to 1 during 10-bit Loopback. The 16-lane core includes logic to force this bit asserted when t2l_config bit-7 or bit-6 is asserted.
LPBK_MODE_SEL	17:16	rst_l	0x0	RW	Loopback Mode Select The LTSSM overrides this value and forces pl_loopback_modep (rtl signal) to 3 during 10-bit Loopback. The 16-lane core includes logic to force these bits asserted to 0x1 when t2l_config bit-7 (force_padlb) is asserted. 00 - SERDES Internal (Serial) Loopback Transmit driver is disabled and the RXLOS is driven by the transmit signal. This is also referred to as EWRAP loopback mode. 01 - SERDES Pad Loopback the transmit driver is enabled and the operation depends on the transmitter termination. 10 - SERDES Reverse Serial Loopback The serial receive data is retransmitted on the serial transmit. 11 - Reverse 10-bit PCS Loopback Transmit received RX input.

Table 1-300 LPU SERDES Glue Config1 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RXLOS_FLTR_EN	15	rst_l	0x1	RW	RxLOS Filtering Enable When 1, enables the SERDES initialization state machine to monitor the filtered RXLOS status. When 0, the status is ignored.
RXLOS_ADJUST	14:12	rst_l	0x1	RW	RXLOS signal threshold adjustment
RXLOS_SMPL_RT	11:8	rst_l	0x0	RW	RXLOS Sample Rate This value is the rate the RXLOS signal is sampled by the RXLOS digital filter.
RXLOS_THRSH_CN	7:0	rst_l	0x19	RW	Threshold Count for RxLOS Filtering This is the number of sample rates RXLOS must be different from the current value before the filtered value changes. When 0, the sample rate only depends on RxLos Sample Rate. If both are zero, the filtered RXLOS will follow the RXLOS value at each clock sample.

1.3.4.202 LPU SERDES Glue Config2 Register (0x006E2808, 0x007E2808 / 0xA16BF1B5)

This register holds SERDES control bits. This register is reset by LPU Reset.rstltssm [4]. See Reset Register.

Table 1-301 LPU SERDES Glue Config2 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
TX_VPULSE_CTL	31:30	rst_l	0x2	RW	TX Voltage Pulse Control
TX_VMUX_CTL	29:28	rst_l	0x2	RW	TX Voltage Mux Control
TX_RISE_FALL	27:25	rst_l	0x0	RW	TX Rise Fall Slew Control
TX_PRE_EMPH	24:22	rst_l	0x5	RW	TX Pre-emphasis Control

Table 1-301 LPU SERDES Glue Config2 Register

Field	Bits	Reset Name	Reset Value	Type	Description
TX_VSWNG_CTL	21:18	rst_1	0xA	RW	TX Voltage Swing Control
TX_PLL_ZERO_CTL	17:16	rst_1	0x3	RW	TX PLL Zero Control
TX_PLL_POLE_CTL	15:14	rst_1	0x3	RW	TX PLL Pole Control
RX_PLL_ZERO_CTL	13:12	rst_1	0x3	RW	RX PLL Zero Control
RX_PLL_POLE_CTL	11:10	rst_1	0x0	RW	RX PLL Pole Control
RX_EQLIZR_CTL	9:6	rst_1	0x6	RW	RX Equalizer Control
OHM_SEL	5	rst_1	0x1	RW	50/75 ohm Select 1 - 50 ohm resistor termination 0 - 75 ohm resistor termination
RTRIMEN	4	rst_1	0x1	RW	rtrim enable When == 0 bits[3:2] and [1:0] are placed on {termr, termt} signals to the SERDES PHY. When == 1 the autotrim {term} output is used.
TX_TERM	3:2	rst_1	0x1	RW	Transmit Termination
RX_TERM	1:0	rst_1	0x1	RW	Receive Termination

1.3.4.203 LPU SERDES Glue Config3 Register (0x006E2810, 0x007E2810 / 0x4401F4)

Electrical Control. This register is reset by LPU Reset.rstltssm [4]. See Reset Register.

Table 1-302 LPU SERDES Glue Config3 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
UNUSED_CNTL3	31:27	rst_1	0x0	RW	Unused Control Bits
OUT_BIAS_CTL	26	rst_1	0x0	RW	Output Bias Control

Table 1-302 LPU SERDES Glue Config3 Register

Field	Bits	Reset Name	Reset Value	Type	Description
TX_RCV_DET	25:24	rst_l	0x0	RW	TX Receiver Detect Window 0x0 52 usec 0x1 53 usec 0x2 54 usec 0x3 55 usec
TX_PLL_HLF_RT_CTL	23	rst_l	0x0	RW	TX PLL Half Rate Control This signal is provided, but normally not connected and unused
TX_PLL_FDBK_DIV	22:20	rst_l	0x4	RW	TX PLL Feedback Divider This value must not be changed since it also controls the Link and Core clock
RX_PLL_HLF_RT_CTL	19	rst_l	0x0	RW	RX PLL Half Rate Control This signal is provided, but normally not connected and unused
RX_PLL_FDBK_DIV	18:16	rst_l	0x4	RW	RX PLL Feedback Divider
BIT_LCK_TM	15:0	rst_l	0x1F4	RW	Bit Lock Time This value determines the delay from a valid receiver input signal or receiver reset to bit lock. This value directly plus the RXLOS filter delay directly effects the N_FTS value which must be sent by this port. The SERDES receiver PLL lock to reference clock is held asserted during this time. If Fast Simulation Mode = 1 (Refer Physical Layer Configuration register), 0x0008 is held preset. The preset overrides the reset value. Default 0x01F4. See RESET_RXPCS & RXPHY_CONF_RCV_RESET(X) in Reset Register and Phy Configuration register. The 2500 bit lock time is 0x00fa. The 5000 bit lock time is 0x01f4.

1.3.4.204 LPU SERDES Glue Config4 Register (0x006E2818, 0x007E2818 / 0x1E848)

This register is reset by LPU Reset.rstltssm [4]. See Reset Register.

Table 1-303 LPU SERDES Glue Config4 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
CFG_UNUSED	31:20	rst_1	0x0	RW	Unused
INIT_TIME	19:0	rst_1	0x1E848	RW	Initialization Time This determines the delay from power-on to receiver PLL lock to reference clock. This value is loaded after Bit Lock Time in SERDES Glue Config3 Register has expired. The preset overrides the reset value. The default value is 0x1e848 (500 usec). The max. timeout value is 0x3ffff (1048 usec). If fast simulation mode = 1 in physical layer configuration register, 0x0009 is preset. The upper 2 bits are not connected to the receiver PLL lock timer logic.

1.3.4.205 LPU SERDES Glue Status Register (0x006E2820, 0x007E2820 / 0xFFFF0000)

This is a read only SERDES Glue Status Register.

Table 1-304 LPU SERDES Glue Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field

Table 1-304 LPU SERDES Glue Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RCV_ELECT_IDLE	31:16	rst_l	0xFFFF	R	Receive Electrical Idle When 1, the receiver Filtered Loss-Of-Signal is asserted. Also asserted when the RXLOS is powered down. Implementation Note: When fewer than 16-lanes are implemented, the unused lanes must have this signal source forced to 1. This ensures the core internal logic does the right thing.
BIT_SYNC_DN	15:0	rst_l	0x0	R	Bit Sync Done Status When 1, the receiver PLL 500 usec timeout completed.

1.3.4.206 LPU SERDES Glue Interrupt and Status Register (0x006E2828, 0x007E2828 / 0x0)

This register is similar to Sun's error status clear register except Bit [31], which is similar to Sun's interrupt status register. This register is reset by LPU Reset.rsterror [8]. See Reset Register.

Table 1-305 LPU SERDES Glue Interrupt and Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
INT_GLOBL_UNMSK	31	por_l	0x0	R	Global Unmasked Interrupt It's 1 when at least one of the unmasked status bits are asserted and the corresponding bit is unmasked in interrupt mask register.
RESERVED	30:24				Reserved field
INT_UNUSED	23:16	por_l	0x0	RW1C	Reserved

Table 1-305 LPU SERDES Glue Interrupt and Status Register

Field	Bits	Reset Name	Reset Value	Type	Description
INT_BYTE_SYNC_STS	15:0	por_1	0x0	RW1C	Byte Sync Status Change When 1, the byte sync status has changed. This status is set when there is a change in the byte sync status which can be read in the byte sync status [15:0] in receive phy status 3 register.

1.3.4.207 LPU SERDES Glue Interrupt and Status Test Register (0x006E2830, 0x007E2830 / 0x0)

This register is similar to Sun's error status set register. Reads of this register return 0.

Table 1-306 LPU SERDES Glue Interrupt and Status Test Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:24				Reserved field
TST_W1S_INT	23:16	rst_1	0x0	RW1S	write 1 to set interrupt
TST_BSSS_INT	15:0	rst_1	0x0	RW1S	Byte Sync Status Change Interrupt Test Bits

1.3.4.208 LPU SERDES Glue Interrupt Mask Register (0x006E2838, 0x007E2838 / 0x80FFFFFF)

This register is similar to Sun's interrupt enable register. Write 0 to enable interrupt and 1 to disable interrupt. This register is reset by LPU Reset.rstltssm [4]. See Reset Register.

Table 1-307 LPU SERDES Glue Interrupt Mask Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
MSK_GLOBL_INT	31	rst_1	0x1	RW	Global Interrupt mask
RESERVED	30:24				Reserved field
MSK_INT	23:0	rst_1	0xFFFF FF	RW	Interrupt Mask

1.3.4.209 LPU SERDES Glue Power Down1 Register (0x006E2840, 0x007E2840 / 0x0)

This register is reset by LPU Reset.rstltssm [4]. See Reset Register.

Table 1-308 LPU SERDES Glue Power Down1 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
TX_PWR_DN	31:16	rst_l	0x0	RW	<p>Transmit Power Down</p> <p>When 1, the transmitter on that lane is powered down and the pu_txlne_activep[15:0] is disabled for that lane. The pt_rcvdet_maskp[15:0] is updated during the LTSSM Detect.Active state for pu_txlane_activep[15:0] which are asserted.</p> <p>Firmware Note: It is recommended these bits never be asserted if the power-on input to the core (t2l_por) can not be asserted by firmware. If this is not the case and these bits are asserted, the transmit skew between lanes may not be met.</p>

Table 1-308 LPU SERDES Glue Power Down1 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RX_PWR_DN	15:0	rst_l	0x0	RW	Receive Lane Power Down During the LTSSM transition to L0 (LTSSM CONFIG_IDLE state & tx_state[IDLE], both pt_rxterm_enable and pt_txlane_enablep are negated on all unused lanes. This forces the transmitters on the unused lanes into electrical idle and the receiver terminators for the unused lanes to be disabled. See GBG_PD2_RX_LANE_PD(X) and GBG_PD2_RXLOS_PD(X). pt_rxterm_enable negated forces RXLOS_PD & RX_LANE_PD asserted. These signals are forced asserted during reset within the PCS selftest logic to disable the receiver terminators. See Transmit Phy Status_2 register.

1.3.4.210 LPU SERDES Glue Power Down2 Register (0x006E2848, 0x007E2848 / 0x0)

This register is reset by LPU Reset.rstltssm [4]. See Reset Register.

Table 1-309 LPU SERDES Glue Power Down2 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:32				Reserved field
PD_UNUSED	31:22	rst_l	0x0	RW	Unused
PWR_DN_CLK_BUF	21	rst_l	0x0	RW	Power Down Clock Buffer This signal is unused in the current design.
PWR_DN_RES_TRIM	20	rst_l	0x0	RW	Power Down Resistor Trim

Table 1-309 LPU SERDES Glue Power Down2 Register

Field	Bits	Reset Name	Reset Value	Type	Description
TX_PLL_PWR_D	19:16	rst_l	0x0	RW	<p>Transmitter PLL Power Down</p> <p>When 1 will power-down each quad transmit PLL. The one used to generate the PHY and Link clock is not connected to the control signal to prevent chip freeze. Once any lane is powered down, the reset_t for all lanes must be asserted for 500 usec during power-up to resync all the transmit FIFO's to minimize the transmit bit skew. The master lane which provides the master 250 MHz clock should not connected to this control since the logic could lock up reset with no clock to remove the reset. When 1, pu_txlane_activep[15:0] is negated in groups of 4. bit[19]=[15:12], 18=[11:8], 17=[7:4], 16=[3:0]. See Transmit Phy Status_2 register.</p> <p>Firmware Note: It is recommended these bits never be asserted if the power-on input to the core (t2l_por) can not be asserted by firmware. If this is not the case and these bits are asserted, the transmit skew between lanes may not be met.</p> <p>Implementation Note: It is a layout requirement for the current implementation to not connect the power-down on the lane which provides the core 250 MHz system clock.</p>

Table 1-309 LPU SERDES Glue Power Down2 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RXLOS_PWR_DN	15:0	rst_l	0x0	RW	Receiver RxLOS Power Down When both the receiver and the rxlos is powered down, the receiver termination is disabled so the other port transmitter will not detect the receiver. This condition is also forced by the PCS hardware during power-on-reset (t2l_por) and on unused lanes. See transmit phy status 2 register.

1.3.4.211 LPU SERDES Glue Config5 Register (0x006E2850, 0x007E2850 / 0x0)

This register is currently unused and may be deleted. This register is reset by LPU Reset.rstltssm [4]. See Reset Register.

Table 1-310 LPU SERDES Glue Config5 Register

Field	Bits	Reset Name	Reset Value	Type	Description
RESERVED	63:0				Reserved field
None					

1.4 Operational Sequences

1.4.1 Power-Up Reset (Hard Reset)

When power is applied, Fire detects the rising edge of the Power-Good (POK) signal from the power supply. Fire resets the entire system and generates J_POR_L and J_RST_L to the processors. There are two variants of the hard reset: long and short.

In order to create short and long “hard” reset, Fire’s XTRNL_RST_L pin is toggled appropriately along with external POK. During POK 0 to 1 transition, XTRNL_RST_L needs to be forced to high to create a long reset, and low to create a short reset.

- For a long reset, J_POR_L deassertion happens 1M clocks after external POK assertion, and J_RST_L deassertion happens ~512K clocks after J_POR_L deassertion.
- For a short reset, J_POR_L gets deasserted 32JBus clocks after external POK assertion, and J_RST_L deassertion happens 33 clocks after J_POR_L deassertion.

Assertion of J_POR_L and J_RST_L are both asynchronous, while deassertions are synchronous. Only the minimum amount of state is defined. The clock ratio will be reset to default. CPU starts fetching the initialization code from the Boot PROM after J_RST_L deassertion.

PCI Express devices and Fire’s internal PCI Express logic gets reset due to external POK, with PCI Express devices and Fire’s internal PCI Express logic coming out of reset after J_RST_L deassertion.

1.4.1.1 Software-Initiated Hard Reset

By setting bit[2] in Fire’s Reset Generation Register, Fire can initiate a hard reset to all processors through J_POR_L and J_RST_L. Note that this software initiated hard reset can only be a long one, assertion and deassertion of J_POR_L and J_RST_L both being synchronous. J_RST_L and J_POR_L need to be asserted for 1M clocks for the processor PLL to lock to new clock ratio. Deassertion of J_RST_L happens ~512K clocks after deassertion of J_POR_L.

PCI Express devices and Fire’s internal PCI Express logic get reset due to software initiated “hard” reset just like hardware initiated “hard” reset.

Note – Hardware initiated Hard reset (from Power_Good) causes PLL reset in Fire, while software initiated Hard Reset does not affect the PLL state. Also in Fire’s PCI Express blocks, the PCI Express PIO AFSR registers get reset by the Hardware initiated Hard reset only, but preserve state across software initiated Hard Reset.

1.4.2 Power-On Reset (Soft Reset)

When power is already on, if the “PB_RST_L” input to Fire gets asserted due to a push-button trigger in the system, Fire initiates a soft reset of the system by asserting J_RST_L to the processors. Assertion of J_RST_L is asynchronous, while deassertion is synchronous. PB_RST_L is level sensitive and must be deasserted before the end of reset, or another reset sequence will be initiated.

When power is already stable and the processor detects a transition on its J_RST_L input pins, it takes a “Soft Reset”. It is similar to a “Hard Reset” except that the on-chip memory controller continues to perform refresh cycles in order to preserve main memory contents, and clock ratio is unaffected. This section describes only the local POR for Jalapeno processors. It is labelled “local” as the rest of the system (like Memory Controller) is not reset.

Soft reset doesn’t reset error reporting/logging/state registers within Fire so that the values in these registers are readable after the soft reset.

There are two variants of the Soft reset: long and short.

To create short and long “soft” reset for the Jalapenos, XTRNL_RST_L pin is toggled appropriately along with PB_RST_L. During PB_RST_L 1 to 0 transition, XTRNL_RST_L needs to be forced to high to create a long reset, and low to create a short reset.

J_RST_L is kept asserted for 1.4M clocks for a long reset, and for 64 clocks for a short reset.

PCI Express devices and Fire’s PCI Express logic get reset due to external PB_RST_L, with PCI Express devices and Fire’s internal PCI Express logic coming out of reset after J_RST_L deassertion.

1.4.2.1 Software-Initiated Soft Reset

By setting bit[0] in the Reset_Gen register of Fire, a Jalapeno processor can also initiate a soft reset via Fire to all processors through J_RST_L. Note that this software initiated soft reset can only be a long one, assertion and deassertion of J_RST_L both being synchronous. J_RST_L needs to be asserted for 5 milliseconds for Jalapeno PLL to lock to new clock ratio.

Fire’s PCI Express logic get reset due to software initiated “soft” reset just like hardware initiated “soft” reset. PCI Express devices may require that a Hot Reset or Link Disable be used.

1.4.2.2 *PCI Express Link Disable*

Any time that a Soft Reset is initiated when a PCI Express link is up, the reset should be proceeded (if possible) by either a Hot Reset or a Link Disable. Moreover, proceeded by a Link Disable is preferred. A link can be disabled prior to the Soft Reset. Link Disable uses the following steps:

1. Set the Disable bit (4) in the TLU Link Control Register.
2. Wait for the link to go down.
3. Initiate Soft Reset.

1.4.2.3 *PCI Express Hot Reset*

Alternatively, there is another way to imitate a Soft Reset when a PCI Express link is up, which is proceeded (if possible) by a Hot Reset. Hot Reset is initiated as follows:

1. Set link to stay in Detect.Quiet by setting bit 8 in the TLU Control Register.

2. Disable active state power management.
3. Write 0x80000401 to the LPU LTSSM Control Register.
4. Wait for the link to go down.
5. Initiate Soft Reset.
6. After Soft Reset, clear bit 8 in the TLU Control Register to initiate link training.

1.4.3 *Externally Initiated Reset (XIR)*

The Jalapeno processor supports the SPARC V9 externally Initiated Reset (XIR) trap. A processor takes this trap when it receives the XIR transaction over JBus, because of signal received on input of Fire or because Soft_XIR bit is set in Reset_Gen Register in Fire. By definition XIR is local to the Jalapeno processor(s). It does not affect the rest of the system. In Jalapeno's case, in order to save some pins on both Fire and Processor module connector, XIR is a transaction on JBus.

Reset due to XIR for the Jalapenos does not initiate fetch of initialization code from Boot PROM, and the memory controller continues to perform refresh cycles in order to preserve main memory contents.

1.4.3.1 Soft XIR

By setting bit[1] in Fire's Reset_Gen Register, a Jalapeno processor can generate an XIR to all processors. Setting that bit causes that Fire to generate an XIR transaction on JBus. The Reset Source register logs the cause of an XIR so that the XIR trap handler can easily identify the source.

Reset due to XIR for the Jalapenos does not initiate fetch of initialization code from Boot PROM, and the memory controller continues to perform refresh cycles in order to preserve main memory contents.

1.4.3.2 *Button/Watchdog XIR*

Fire supports generating an XIR transaction to all processors triggered from an external source. Expected usage of this feature is for a Button XIR and/or a System Watchdog Reset. Bit[5] of the Reset Source register is set to one when a externally triggered XIR is generated by Fire. This allows the trap handler to identify the cause of the XIR.

The XTRNL_RESET_L pin (used to force an XIR) is edge sensitive, a high to low transition causes and XIR to be sent. Its level sensitive when used to control reset timing.

1.4.4 *Reset Summary*

The table below shows how Hard/Soft reset effects Fire and how these terms map to PCI-Express reset terms.

Table 1-311 Reset Terminology

Function	Sun/Fire ASIC	PCI-Express Spec
Reset of all chip state including errors	Hard	Cold
Reset of all non-error chip state	Soft	Warm

The Table below summarizes how resets (and XIR) can be initiated. There are hardware events that can cause both Soft/Hard resets and XIR as well as software initiated actions to cause all of these.

Table 1-312 Reset Sources and effects

Source	Hard Reset	Soft Reset	XIR
Hardware	Power Good	Push button (PB_RESET)	Push Button (XTRNL_RESET)
		Fatal Error (DOK_ON 4 times on JBUS)	
Software	CSR Write (Hard Reset Req)	CSR Write (Soft Reset Req)	CSR Write (XIR Request)

1.4.5 POR & Warm Reset Initialization

This section provides examples in the initialization of specific key registers in Fire.

For these examples Fire's AgentID is 0x1C. Therefore, Fire's registers are located at JBus address 0x400.0E00.0000 and the 64 GByte non cacheable region starts at 0x7C0.0000.0000

1.4.5.1 Example Address Initialization

This example demonstrates initialization of Fire's JBus parity and address mapping registers.

- JBus Initialization
 - Enable JBus Parity Control <= 0x8000.0000.0000.0000
- Setup Mapping Registers
 - Leave reset values in Ebus base and mask
 - PCIE-A Cfg/IO Base <= 0x8000.0009.0000.0000
 - PCIE-A Cfg/IO Mask <= 0x0000.000F.E000.0000
 - PCIE-A Mem32 Base <= 0x8000.0008.0000.0000
 - PCIE-A Mem32 Mask <= 0x0000.000F.8000.0000
 - PCIE-A Mem64 Base <= 0x8000.0000.0000.0000
 - PCIE-A Mem64 Mask <= 0x0000.000C.0000.0000
 - PCIE-A Mem64 PCIE Offset <= 0xFFFF.FFFC.0000.0000
 - PCIE-B Cfg/IO Base <= 0x8000.0009.2000.0000
 - PCIE-B Cfg/IO Mask <= 0x0000.000F.E000.0000
 - PCIE-B Mem32 Base <= 0x8000.0008.8000.0000
 - PCIE-B Mem32 Mask <= 0x0000.000F.8000.0000
 - PCIE-B Mem64 Base <= 0x8000.0004.0000.0000

- PCIE-B Mem64 Mask $\leq 0x0000.000C.0000.0000$
- PCIE-B Mem64 PCIE Offset $\leq 0xFFFF.FFFC.0000.0000$

As a result of these settings, the address mapping of the sub-regions will be as follows:

Ebus $\Rightarrow 0x7CF.F000.0000 - 0x7CF.F7FF.FFFF$ (128 MBytes)
PCIE-B Cfg/IO $\Rightarrow 0x7C9.2000.0000 - 0x7C9.3FFF.FFFF$ (512 MBytes)
PCIE-A Cfg/IO $\Rightarrow 0x7C9.0000.0000 - 0x7C9.1FFF.FFFF$ (512 MBytes)
PCIE-B MEM32 $\Rightarrow 0x7C8.8000.0000 - 0x7C8.FFFF.FFFF$ (2 GBytes)
PCIE-A MEM32 $\Rightarrow 0x7C8.0000.0000 - 0x7C8.7FFF.FFFF$ (2 GBytes)
PCIE-B MEM64 $\Rightarrow 0x7C4.0000.0000 - 0x7C7.FFFF.FFFF$ (16 GBytes)
PCIE-A MEM64 $\Rightarrow 0x7C0.0000.0000 - 0x7C3.FFFF.FFFF$ (16 GBytes)

1.4.5.2 Example MMU Initialization

1.4.5.2.1 MMU Bypass

- PCIE-A MMU Control $\leq 0x0000.0000.0000.0002$

1.4.5.2.2 MMU TSB Initialization

- Set TSB base Address
 - PCIE-A MMU TSB Control $\leq \langle \text{TSB Physical Address} \rangle$
- Enable MMU and Bypass
 - PCIE-A MMU Control $\leq 0x0000.0000.0000.0703$

1.4.5.2.3 MMU TTE Cache Initialization (optional)

To preload the TTE Cache which can be done to avoid creating a translation storage buffer in main memory, use the following steps:

1. Set the Cache Mode in the MMU Control and Status Register to 0b00.
2. Write the eight aligned and consecutive TTE entries in the translation data buffer.
3. Write the virtual tag corresponding to the eight TTE entries.
 - a. The CNT field should be written with 0x000
 - b. The TAG field should be written with the virtual address bits. If the page size is 64 K then bits 18:16 of the TAG must be written as 0b000.
 - c. The VLD field should be written with 0b1.
4. Write the physical tag corresponding to the virtual tag. This is only required if snoop or address invalidation is required.
 - a. The TAG field is written with the physical address of the TSB entries.
 - b. The VLD field should be written with 0b1.

1.4.5.2.4 MMU TTE Cache Invalidation

The MMU TTE Cache Invalidate Register can be used to invalidate specific entries in the cache. To invalidate the entire cache write 0xFFFFFFFFFFFFFFFF. This can be done at any time. To invalidate less than the entire cache, use of the cache must first be suspended. The entire procedure is as follows:

1. Suspend the MMU by setting the PD bit in the MMU Control and Status Register.
2. Wait until the MMU is quiescent by polling the TLP and TIP bits in the previous register until both are clear.
3. Write to the MMU TTE Cache Invalidate Register with whatever pattern is required.
4. Resume by clearing the PD bit.

1.4.5.3 Example PCI-E Link Initialization

- Enable ASPM
 - PCIE-A TLU Link Control <= 0x0000.0000.0000.0003
- Use HW auto-config timers
 - PCIE-A LPU Link Layer Config <= 0x0000.0000.0002.0114
- Manually configure timers
 - PCIE-A LPU Link Layer Config <= 0x0000.0000.000b.0114

1.4.5.4 Example Interrupt Mapping Initialization

- Interrupt Mapping (MDO_MODE,V,T_JPID=3,INT_CNTRL_NUM=1)
 - PCIE-A Interrupt Mapping <= 0x8000.0000.8C00.0040
- Initialize Interrupt Mondo Data 1 Reg
 - PCIE-A Interrupt Mondo Data 1 Reg <= <Mondo Data>
- Initialize Event Queue Base Address Reg
 - PCIE-A Event Queue Base Address <= <EQ Base Address>
- Clear Event Queue Head Reg
- Clear Event Queue Tail Reg
- Event Queue Control Set Register (EQ 0 register)
 - PCIE-A Event Queue Control Set <= 0x0000.1000.0000.0000
- MSI 64-bit Address Register (initialize MSI 64-bit Address)
 - PCIE-A MSI 64-bit Address <= <MSI 64-bit Address>
- MSI Mapping Register 0 (map MSI 0 to Event Queue)
 - PCIE-A MSI Mapping Register 0 <= 0x8000.0000.0000.0000

1.4.5.5 Link Training

The link does not proceed to train unless “Remain in Detect.Quiet” bit [8] in TLU Control Register is 1'b0, where the bit is default to 1'b1. Therefore, after finish all the CSR initialization, SW must write to TLU Control Register to set bit [8] to 1'b0.

After “Remain in Detect.Quiet” bit is cleared, it takes a max. 80 ms to train a link.

To prevent the link re-train automatically after it's down, SW must set bit [8] in TLU Control Register back to 1'b1 after it receives a link up interrupt.

1.4.5.6 PCI Express Configuration

Fire does not return unsupported request status when issuing configuration type 0 commands directed at devices other than device zero. Configuration software must be aware of this and only use a single device id (0) for devices which are directly connected to a Fire port.

For systems which can issue 64-byte block reads to the PCI Express fabric, the default initial credits which are advertised must be reduced. The PDC field of the TLU Ingress Credits Initial Register must be reduced from the default of 0x0C0 to 0x0BC. This must be done before the final reset before the link is allowed to train.

1.4.6 Estar Sequence

In systems where Estar is supported, the ILU Device Capabilities register must be written to 1 (in Estar Mode) during Initialization for system Estar supported mode. This register get reset to be in non-Estar mode on both hard and soft resets and therefore must be re-written to be in Estar mode after every reset.

Note – Writing the ILU Device Capabilities register changes the behavior of a Fire internal interface slightly, causing a slight increase, on the order of ~2%, to the latency of DMA Reads in Fire.

All JBus Devices, in a system, must have their JBus Energy Star Control registers written to the same target frequency (full, 1/2 or 1/32 speed), this includes the JBus Energy Star Control Register in Fire (for both boot path and non boot path Fires).

Note – In systems with Bells (JBus repeater chips) GPIOs external to bells must also be written with the proper target speed values. Please refer to system documents for the location of these GPIOs.

After all JBus Energy Star Control Registers are written, the JBus Change Initiation Control Register in the boot path Fire must be written. Writing this register initiates a JBus Speed Change sequence that actually causes the JBus speed to change. See the CSR descriptions of these registers for their programming values.

Speed change between 1/32 mode and full speed mode (in either direction) must go through 1/2 mode. No direct change between full mode and 1/32 mode is allowed.

The above sequence should be followed for both speed decreases and speed increases.

In systems using the I2C controllers in fire, changing the JBus clock will change the frequency of the I2C bus (the reference clock for the I2C controller is the EStar controlled version of the JBus Clock). If it is desired that the I2C bus maintain similar frequencies regardless of the JBus clock frequency, the I2C bus clock divisor must be changed when the JBus clock frequency is changed.

If the JBus Clock Frequency is decreasing, the JBus clock frequency **MUST** be decreased **BEFORE** the I2C Clock Divisor is modified. If the JBus clock frequency is increasing, the I2C Clock Divisor **MUST** be modified **BEFORE** the JBus clock frequency is increased. By following this behavior, it is ensured that the I2C bus frequency never is higher than is maximum specified value.

1.4.7 Internal Loopback Program Instructions

The internal loopbacks are loopbacks of the PCIE interface. The transactions are initiated on JBus and turned around on PCIE interface.

To make the internal loopback work, some special CSR configurations are needed. Here are the program instructions to configure Fire for internal loopbacks. The instructions work for both port A and port B. Thus, port A is used as an example in the instructions. Moreover, the example given below is for MMU bypass mode.

1. Set the digital, ewrap or pad loopback enable bit. The CSR is "TLU Control Register" and its offset is 0x680000. The CSR accesses are read, modify and write back to set bits [7:5] as 3'b001 for digital mode, 3'b010 for ewrap mode and 3'b100 for pad mode.

Note – Set the three bits [7:5] in "TLU Control Register" mutual exclusive when enabling an internal loopback mode.

2. Issue a soft reset to take the configured loopback mode effective in LPU. The CSR is "Reset Generation Register" and its offset is 0x417010. The CSR access is a write with data 64'h1.
3. If it's ewrap or pad loopback mode, set receiver detect bypass mode bit. The CSR is "LPU Physical Layer Configuration Register" and its offset is 0x6E2600. The CSR accesses are read, modify and write back to set bit[3] to 1'b1. The operation is to force the receiver detect value to be the maximum width select value.
4. If it's ewrap or pad loopback mode, set bit[8] to 1'b0 in "TLU Control Register" to let LTSSM leave the Detect.Quiet state.
5. Check if the link is up by issuing a PIO read. The CSR is "TLU Status Register" and its offset is 0x00680008. If the value is 0x4, the link is up.
6. If the link is down, repeat the previous step until the link is up.
7. Do normal CSR initiation sequence
 - a. Initiate all the base and mask registers, which are
 - i. Ebus Offset Base/Mask Register
 - ii. PCIE-A/B Mem32 Offset Base/Mask Register
 - iii. PCIE-A/B Cfg/IO Offset Base/Mask Register
 - iv. PCIE-A/B Mem64 Offset Base/Mask Register
 - b. Enable all the interrupt bits, but leave PEC UE, CE, and OE interrupt enable bit off in the "PEC Core and Block Interrupt Enable Register". The PEC UE, CE, and OE bits are bit[2:0] in the register. The reason to leave these bits off is due to some expected LPU behaviors. However, it's optional to enable the interrupt bits for loopbacks. During normal loopback operations, any interrupts are unexpected when the interrupts mentioned above are enabled.
 - c. If the interrupts mentioned above are enabled, the initiations for the following CSRs are needed to generate mondo.
 - i. Interrupt Retry Timer Register, offset 0x601a00

- ii. Interrupt Mondo Data 0 Register, offset 0x62C000
- iii. Interrupt Mondo Data 1 Register, offset 0x62C008
8. Set DMA bypass enable bit. The CSR is "MMU Control and Status Register" and its offset is 0x640000. The bypass enable bit is bit[1].
9. Set Mem64 PCIE offset to match the DMA bypass address in Fire. The CSR is "Mem 64 PCIE Offset Register" and its offset is 0x634018. The CSR access is a write with data 64'hFFFC_0000_0000_0000.
10. From this point on, Fire is ready to function on loopbacks.

1.4.8 The Drain State

The purpose of Drain State is to unstage Fire's egress pipeline from JBC to PEC. This is required to enable internal PIOs issued by software to propagate to the CSR ring. In JBC, external and internal PIOs are stored in a single queue. If the pipeline stalls, internal PIOs could be blocked in the queue. Software would then be unable to read/clear the internal CSRs. Drain State alleviates this blockage and allows CSR access to Software.

There are four scenarios, which will get Fire into the Drain State:

- Ingress Header Buffer (IHB) parity error
- Egress Header Buffer (EHB) parity error if EHP_P and EHP_S bits are enabled in TLU Other Event Log Enable Register
- Egress Data Buffer (EDB) parity error if EDP_P and EDP_S bits are enabled in TLU Other Event Log Enable Register
- Link down transition if LDN_P and LDN_S bits are enabled in TLU Other Event Log Enable Register

When Fire is in the Drain State, Fire:

- short-circuits PIO reads by manufacturing corresponding PIO completions with "Unsupported Request" status
- silently drops PIO writes
- stops receiving PCI-E traffic
- silently drops outgoing DMA completions

Note – The LTSSM may stay in the normal operational (L0) state when Drain State is caused by an IHB, and/or EHB, and/or EDB parity error.

The reason for completing PIO reads is to free the processors who issued the PIOs from waiting for their PIO completions and be able to issue new PIOs to internal CSRs.

When any one of the above four scenarios happens, the detector of the error will set its corresponding CSR error status bit if enabled and request for interrupt if enabled. Here are the steps SW will take when it receives the interrupt:

- PIO reads to the CSRs to identify the source of the problem.
- A PIO write to the CSR to clear the error status bit.

- Reset the chip if Drain State is caused by an IHB, and/or EHB, and/or EDB parity error, which are not recoverable since the PCI-E flow control credits are lost due to the header or data loss when a parity error occurs.
- Follow the procedures in the section of "Re-train a Link After it is Down" to get out of the drain state if Drain State is caused by the link down transition.

Note – The Drain bit in "TLU Status Register" is set by HW only when the Drain State is caused by (1) Egress Header Buffer (EHB) parity error, or (2) Egress Data Buffer (EDB) parity error, or (3) Link down transition.

1.4.9 *Re-train a Link After it is Down*

The following is the procedure if software wishes to attempt to bring a PCI Express link up after it has gone down. Software must quiesce the system by executing the following three steps in order:

1. Software must shutdown/suspend any process which may make PIO requests to the PCI Express link in question.
2. Next, to ensure that no requests are outstanding, software issues 16 read requests to the link. All of these will complete with bus error status.
3. Next, software must ensure that no outstanding DMA read requests are outstanding by checking that no entries are on the Transaction Scoreboard.

Once the system is quiesced, software may bring up the link by executing the following three steps in order:

1. If another process has not already done so, the Link Down bit in the TLU Other Event Clear Status Register must be cleared by writing a one to that bit.
2. Next, the Drain bit in the TLU Status Register must be cleared by writing a one to that bit.
3. Next, the Remain in Detect.Quiet bit in the TLU Control Register must be cleared by writing a zero to that bit. The LTSSM must be in the state Detect.Quiet before this bit is cleared. This is indicated by the LTSSM_STATE field in the LPU LTSSM Status1 Register having the value of 0x00.

Once the link is up, the Remain in Detect.Quiet bit in the TLU Control Register should be set.

1.4.10 *PCI-E Device Power Off and On*

Fire does not support the PCI Express link power management state L2 and power off to Fire is not supported. Thus, LTSSM L2 and L3 states are not implemented in Fire.

Fire enters L1 state not only when it is directed to transition to L1 state, but also when it is directed to transition to L2/L3 Ready. However, from the L1 state, the state in which Fire transitions to depends on how it was directed to the L1 state. When the conditions exist to exit L1 state:

- Fire transitions to Recovery state from L1 state if the L1 state results from SW directed entering L1 state, such as that SW directs all functions of a downstream component to D3(hot).
- Fire transition to Detect state from L1 state if the L1 state results from SW directed entering L2/L3 Ready, such as that SW writes to the PTO bit in the PME Turn Off Generate Register.

After SW directs Fire transition to L2/L3 Ready, SW can notify the power supply to power off the downstream device if the event queue write for the PM_TO_Ack message is received and Fire's LTSSM state is in L1 state by reading the LPU LTSSM Status1 Register. Please note that Fire stays in L1 state when the downstream device is power off and the link is still considered **up** from Fire's perspective.

To bring the link back after the downstream device powered off, SW first notifies power supply to power on the downstream device. When the downstream device is powered back on, Fire transitions from L1 to Detect state, which causes the link go **down**. If primary and secondary link down transition event (LDN_P and LDN_S) log are enabled in TLU Other Event Log Enable Register (which is the case in normal operation), Fire goes to the Drain State due to the link down transition. Please refer to Section 1.4.8, "The Drain State," on page 1-371 for details about the Drain State. Please follow the procedures in Section 1.4.9, "Re-train a Link After it is Down," on page 1-372 to bring the link back on.

Once the link is up, the Remain in Detect.Quiet bit in the TLU Control Register should be set, the log for primary and secondary link down event (LDN_P and LDN_S) in TLU Other Event Log Enable Register should be enabled.

1.4.11 *SERDES Electrical SERDES Configuration*

Fine tuning of the PCI-E electrical levels can be accomplished through the SERDES Glue Config1, Config2, and Config3 registers. The default values of these registers in Fire 2.0 is adequate for all Fire platforms, though a more perfect 'eye' diagram can be achieved by adjustments to these values.

Fire 2.0 default value of LPU SERDES Glue Config2 Register is: 0xA16BF1B5.

Ontario/Erie signal integrity work determined the optimum value of this register to be: 0xA06BF1B5.

The single difference between the reset value of the register and the recommended register setting for Ontario and Erie is the emphasis setting. The reset value has a moderate amount of emphasis (101) and due to the short PCI-E trace lengths on Ontario/Erie a small amount of emphasis (001) was sufficient to equalize the swing of the transition bits and non-transition bits at the receiver.

For maximum margin, some of these register settings should be tweaked (EMPH, VTXLO, etc.) depending on the system topology. Note that each platform is responsible for its own tuning.

Note – Changing the value of the TX_PLL_ZERO_CTL (TZ) and/or TX_PLL_POLE_CTL (TP) fields of the LPU SERDES Glue Config2 Register while Fire's PCI-E link is trained will cause Fire to transmit garbage on the link for a short period

of time. Downstream devices may react to this bad data, therefore if these fields are to be adjusted it should be done when the link is not trained (i.e. in LTSSM Detect Quiet state) or the downstream device error state should be cleaned up if necessary.

The GB signals mentioned are mapped to three registers, which are LPU SERDES Glue Config1 register, LPU SERDES Glue Config2 register, and LPU SERDES Glue Config3 register. The following table lists the mappings between the SERDES signals and the PRM register fields.

Table 1-313 Mapping table of SERDES Signals to PRM Register Fields

Serdes Signal Reference	PRM Register Reference	Default Setting
VTXLO[0:3]	LPU SERDES Glue Config2[21:18]	1010
EMPH[0:2]	LPU SERDES Glue Config2[24:22]	101
VMUXLO[0:1]	LPU SERDES Glue Config2[29:28]	10
VPULSELO[0:1]	LPU SERDES Glue Config2[31:30]	10
RISEFALL[0:2]	LPU SERDES Glue Config2[27:25]	000
RXEQ[0:3]	LPU SERDES Glue Config2[09:06]	0110
LOSADJ[0:2]	LPU SERDES Glue Config1[14:12]	001
TP[0:1]	LPU SERDES Glue Config2[15:14]	11
TZ[0:1]	LPU SERDES Glue Config2[17:16]	11
RP[0:1]	LPU SERDES Glue Config2[11:10]	00
RZ[0:1]	LPU SERDES Glue Config2[13:12]	11
BIASCNTRL	LPU SERDES Glue Config3[26]	0

1.4.12 DR

Fire does not support dynamic re-configuration.

Note – Fire does support non-destructive JTAG CSR accesses. Refer to the *CSR Access via JTAG* section in the Fire Chip Specification.

1.4.12.1 Remapping of functionality

Fire does not support domainning.

1.4.12.2 Removal from active operation

Fire does not support dynamic removal of itself.

1.4.12.3 Interrupt Redirection

invalidate_interrupt_mapping_state(ino);

```
while (interrupt_state(ino) != IDLE)
    ;
set_cpuid_in_interrupt_mapping(ino, cpuid);
validate_interrupt_mapping_state(ino);
```

1.4.13 Error Event Summary

Error events are summarized in the following table.

In each error status clear register, primary errors correspond to the first occurrence of the errors in that register and the associated information are captured in the associated error log register(s). Secondary errors are cumulative, this means that more than one secondary error bit can be set if two or more errors in that register occurred.

Note – Theoretically Fire can get multiple errors from within the same group but only if they are asserted in the **same** cycle, though realize that the source transaction for these multiple errors would be the same so there's no need for hardware to prioritize them or the logging of their data.

Note – All individual errors are maskable at the block level. All error interrupts are maskable at the block level. All errors from a given block are maskable at the core level. And each core's errors can be individually masked. See Error Methodology Appendix for a full description.

Note – Driving a system/soft reset in response to a given fatal error is maskable at the JBC core level by use of the JBC Fatal Reset Enable Register. Fire will not recover from a fatal error, however, regardless of these settings.

Table 1-314 Error Source and Handling Table (Sheet 1 of 22)

Event Detector: (If an event is transaction specific, note dependencies in parentheses)	Information Captured: (e.g. Error Status bit, syndrome, bad packet, failing chip indicator, etc. and the CSR register.)	Reporting Mechanism: (e.g. error line, bus interrupt and priority, etc.)	Impact: (e.g. Corrected, Retried, Continues in degraded mode, fatal to chip, Fatal to Domain, Fatal to multiple domains, etc.)
JBC ERRORS			
Merge Buffer Parity Error (rd_buf) (parity error detected when reading from the rd_buf RAM of the MB. Can be caused by copybacks and Fire's WRB)	<p>A maskable error is logged in the JBC Error Status Clear register bit 24/56 (primary/secondary).</p> <p>The following fields are logged in the Merge Transaction Error Log Register:</p> <ul style="list-style-type: none"> * Address of the original partial write txn that wrote data into the rd_buf entry. * jbc_tag of the original partial write transaction * Quad word offset - The data beat containing the error. * j_adtype[5:0] of the DMA transaction or copyback request. 	A maskable interrupt is generated.	* The data is written or copied back to the requesting device with a UE error set on the 16 byte chunk(s) of data containing the parity error.
Merge Buffer Parity Error (wr_buf) (parity error detected when reading from the wr_buf RAM of the MB. Can be caused by Fire issued WRB, WRBC, WRIS, as well as PIO 16/64 read returns) (Since Fire only checks data parity at the end, this error may be in PEC MB wr_buf RAM, DMC DIU RAM, PEC IDB RAM, or PEC Ingress interface parity error if it's associated with payload)	<p>A maskable error is logged in the JBC Error Status Clear register bit 23/55 (primary/secondary).</p> <p>The following fields are logged in the Merge Transaction Error Log Register:</p> <ul style="list-style-type: none"> * Address of the original partial write txn that wrote data into the rd_buf entry. * jbc_tag of the original partial write transaction * Quad word offset - The data beat containing the error. * j_adtype[5:0] of the DMA transaction or copyback request. 	A maskable interrupt is generated.	* The data is written or copied back to the requesting device with a UE error set on the 16 byte chunk(s) of data containing the parity error.
Merge Buffer Address Parity Error (addr_tag) (parity error detected when reading from the addr_tag RAM in the cmd queue.)	<p>A maskable error is logged in the JBC Error Status Clear register bit 25/57 (primary/secondary).</p> <p>The following fields are logged in the Fatal Error Log Register 1:</p> <ul style="list-style-type: none"> * Address of the original DMA txn * jbc_tag of the original transaction * j_adtype[5:0] of the DMA transaction. 	<p>A maskable interrupt is generated.</p> <p>A maskable system reset is generated.</p>	* This is a FATAL error and will result in a system reset (4 DOK_ONs) unless this behavior is masked by system software. Fire will not recover from this error, regardless of whether a system reset occurs or not.

Table 1-314 Error Source and Handling Table (Sheet 2 of 22)

Event Detector: (If an event is transaction specific, note dependencies in parentheses)	Information Captured: (e.g. Error Status bit, syndrome, bad packet, failing chip indicator, etc. and the CSR register.)	Reporting Mechanism: (e.g. error line, bus interrupt and priority, etc.)	Impact: (e.g. Corrected, Retried, Continues in degraded mode, fatal to chip, Fatal to Domain, Fatal to multiple domains, etc.)
UE Asynchronous Fault Error (RDS, RDD, RDSA, RDO issued by Fire. This error is specific to the read return data cycles of these txns)	<p>A maskable error is logged in the JBC Error Status Clear register bit 22/54 (primary/secondary).</p> <p>The following fields are logged in the JBusint In Transaction Error Log Register:</p> <ul style="list-style-type: none"> * Address of the original read txn. * j_adtype[5:0] (Fire's transid) captured during the first cycle of the data return. * Quad word offset - The data beat containing the error. <p>The following fields are logged in the JBusint In Transaction Error Log Register 2:</p> <ul style="list-style-type: none"> * Last 8 cycles worth of JBus arb winners * j_req line state when error occurred * j_pack lines when error occurred. 	A maskable interrupt is generated.	<p>* For RDS, RDD, and RDSA, the read transaction is completed in error to the DMC core with 4 data beats.</p> <p>* For an RDO, processing will continue as normal and a write completion will eventually be sent to the DMC core; Fire will retain ownership of the cacheline so the WRB will occur normally but with a UE error indicated on the bad data. The actual data written or copied back depends on the cs2mb_ue_prop_mode bit.</p> <p>If 1'b1: Regular merged data is returned</p> <p>If 1'b0: Original RDO read data is returned untouched</p>
CE Asynchronous Fault Error (RDS, RDD, RDSA, RDO issued by Fire. This error is specific to the read return data cycles of these txns)	<p>A maskable error is logged in the JBC Error Status Clear register bit 21/53 (primary/secondary).</p> <p>The following fields are also logged in the JBusint In Transaction Error Log Register:</p> <ul style="list-style-type: none"> * Address of the original read txn. * j_adtype[5:0] (Fire's transid) captured during the first cycle of the data return. * Quad word offset - The data beat containing the error. <p>The following fields are logged in the JBusint Transaction Error Log In Register 2:</p> <ul style="list-style-type: none"> * Last 8 cycles worth of JBus arb winners * j_req line state when error occurred * j_pack lines when error occurred. 	A maskable interrupt is generated.	* The transaction will complete as normal.

Table 1-314 Error Source and Handling Table (Sheet 3 of 22)

Event Detector: (If an event is transaction specific, note dependencies in parentheses)	Information Captured: (e.g. Error Status bit, syndrome, bad packet, failing chip indicator, etc. and the CSR register.)	Reporting Mechanism: (e.g. error line, bus interrupt and priority, etc.)	Impact: (e.g. Corrected, Retried, Continues in degraded mode, fatal to chip, Fatal to Domain, Fatal to multiple domains, etc.)
JBus Timeout Error (RDS, RDD, RDSA, RDO issued by Fire. This error occurs when Fire receives a read data error packet in response to one of its DMA reads)	<p>A maskable error is logged in JBC Error Status Clear register bit 20/ 52 (primary/secondary).</p> <p>The following fields are logged in the JBusint In Transaction Error Log Register:</p> <ul style="list-style-type: none"> * Address of the original read txn. * j_adtype[5:0] (Fire's transid). <p>The following fields are logged in the JBusint In Transaction Error Log Register 2:</p> <ul style="list-style-type: none"> * Last 8 cycles worth of JBus arb winners * j_req line state when error occurred * j_pack lines when error occurred. 	A maskable interrupt is generated.	<ul style="list-style-type: none"> * For RDS, RDD, and RDSA, the read transaction is completed in error to the DMC core without data beats. * For an RDO, processing will continue as normal and a write completion will eventually be sent to the DMC core; The WRB will occur normally with a UE error indicated on all four data beats. The WRB will contain the original write data untouched and not merged data since no valid read data was returned.
JBus Bus Error (RDS, RDD, RDSA, RDO issued by Fire. This error occurs when Fire receives a read data error packet in response to one of its DMA reads)	<p>A maskable error is logged in JBC Error Status Clear Register bit 19/ 51 (primary/secondary).</p> <p>The following fields are logged in the JBusint In Transaction Error Log Register:</p> <ul style="list-style-type: none"> * Address of the original read txn. * j_adtype[5:0] (Fire's transid). <p>The following fields are logged in the JBusint In Transaction Error Log Register 2:</p> <ul style="list-style-type: none"> * Last 8 cycles worth of JBus arb winners * j_req line state when error occurred * j_pack lines when error occurred. 	A maskable interrupt is generated.	<ul style="list-style-type: none"> * For RDS, RDD, and RDSA, the read transaction is completed in error to the DMC core without data beats. * For an RDO, processing will continue as normal and a write completion will eventually be sent to the DMC core; The WRB will occur normally with a UE error indicated on all four data beats. The WRB will contain the original write data untouched and not merged data since no valid read data was returned.

Table 1-314 Error Source and Handling Table (Sheet 4 of 22)

Event Detector: (If an event is transaction specific, note dependencies in parentheses)	Information Captured: (e.g. Error Status bit, syndrome, bad packet, failing chip indicator, etc. and the CSR register.)	Reporting Mechanism: (e.g. error line, bus interrupt and priority, etc.)	Impact: (e.g. Corrected, Retried, Continues in degraded mode, fatal to chip, Fatal to Domain, Fatal to multiple domains, etc.)
JBus Unmapped Error (PIO writes to invalid addresses within Fire's address range detected by the j_jbusint block including NCBWR to 8 MB space, non-aligned NC(B)WR to TSB flush register, NC(B)WR to cacheable Fire space; DMA read data error packet indicating an unmapped error)	<p>A maskable error is logged in JBC Error Status Clear Register bit 18/50 (primary/secondary).</p> <p>The following fields are logged in the JBusint In Transaction Error Log Register:</p> <ul style="list-style-type: none"> * Address of the original txn. * j_adtype[5:0] (source device's transid). <p>The following fields are logged in the JBusint In Transaction Error Log Register 2:</p> <ul style="list-style-type: none"> * Last 8 cycles worth of JBus arb winners * j_req line state when error occurred * j_pack lines when error occurred. 	A maskable interrupt is generated.	<ul style="list-style-type: none"> * For DMA read data returns for RDS, RDD, and RDSA, the read transaction is completed in error to the DMC core without data beats. * For an RDO, processing will continue as normal and a write completion will eventually be sent to the DMC core; The WRB will occur normally with a UE error indicated on all four data beats. The WRB will contain the original write data untouched and not merged data since no valid read data was returned. * Unmapped PIO writes are dropped.
Invalid JBus Port Error (All Fire issued transactions to nonexistent devices with the exception of DMA reads which will be handled by the DMA Read Response Timeout.)	<p>A maskable error is logged in the JBC Error Status Clear register bit 17/49 (primary/secondary).</p> <p>The following fields are logged in the JBusint Out Transaction Error Log Register:</p> <ul style="list-style-type: none"> * Address of the txn. * j_adtype[5:0] (Fire's transid) <p>The following fields are logged in the JBusint Out Transaction Error Log Register 2:</p> <ul style="list-style-type: none"> * Last 8 cycles worth of JBus arb winners * j_req line state when error occurred * j_pack lines when error occurred. 	A maskable interrupt is generated.	<ul style="list-style-type: none"> * For Fire issued transactions, the transaction is issued on JBus normally. Thus, DMA writes will generate an acknowledgement to the j_dmcint block.

Table 1-314 Error Source and Handling Table (Sheet 5 of 22)

Event Detector: (If an event is transaction specific, note dependencies in parentheses)	Information Captured: (e.g. Error Status bit, syndrome, bad packet, failing chip indicator, etc. and the CSR register.)	Reporting Mechanism: (e.g. error line, bus interrupt and priority, etc.)	Impact: (e.g. Corrected, Retried, Continues in degraded mode, fatal to chip, Fatal to Domain, Fatal to multiple domains, etc.)
Illegal Cache Install State (RDS, RDD, RDO issued by Fire. This error is specific to the read return data cycles of these txns)	<p>A maskable error is logged in JBC Error Status Clear Register bit 16/48 (primary/secondary).</p> <p>The following fields are logged in the JBusint In Transaction Error Log Register:</p> <ul style="list-style-type: none"> * Address of the original read txn. * j_adtype[5:0] (Fire's transid). <p>The following fields are logged in the JBusint In Transaction Error Log Register 2:</p> <ul style="list-style-type: none"> * Last 8 cycles worth of JBus arb winners * j_req line state when error occurred * j_pack lines when error occurred. 	A maskable interrupt is generated.	<p>* For an RDO, processing will continue and a write completion will eventually be sent to the DMC core; Fire will retain ownership of the cacheline so the WRB will occur normally. For an RDD and RDS, processing will also continue. We do this because an illegal cache install state can only occur if a parity error occurs on JBus (either that or the CPU is seriously damaged in which case the whole system has been comprised) so in this case, the read transaction will just go through the normal data parity error processing channels in addition to having an illegal cache state error being logged.</p>
Control (j_par) Parity Error (This error is associated with the JBus signals that are looked at in the calculation of the j_par signal)	<p>A maskable error is logged in JBC Error Status Clear Register bit 15/47 (primary/secondary).</p> <p>The following fields are logged in the Fatal Error Log Register 2:</p> <ul style="list-style-type: none"> * Last 8 cycles worth of JBus arb winners * j_req line state when error occurred * j_pack lines when error occurred. 	<p>A maskable interrupt is generated.</p> <p>A maskable system reset is generated.</p>	<p>* This is a FATAL error and will result in a system reset (4 DOK_ONs) unless this behavior is masked by system software. Fire will not recover from this error, regardless of whether a system reset occurs or not.</p>
Address Parity Error (all address cycle packets. j_adtype[7:6] == 0x3. Also, a parity error in j_adp[3] during the first cycle of a RDR64 is treated as an address parity error)	<p>A maskable error is logged in JBC Error Status Clear Register bit 14/46 (primary/secondary).</p> <p>The following fields are logged in the Fatal Error Log Register 1:</p> <ul style="list-style-type: none"> * Address of the original txn. * j_adtype[5:0] (Fire's transid). <p>The following fields are logged in the Fatal Error Log Register 2:</p> <ul style="list-style-type: none"> * Last 8 cycles worth of JBus arb winners * j_req line state when error occurred * j_pack lines when error occurred. 	<p>A maskable interrupt is generated.</p> <p>A maskable system reset is generated.</p>	<p>* This is a FATAL error and will result in a system reset (4 DOK_ONs) unless this behavior is masked by system software. Fire will not recover from this error, regardless of whether a system reset occurs or not.</p> <ul style="list-style-type: none"> * Transaction is dropped by the j_jbusint block * Note that processing of new JBus txns does continue but there's no way to recover from the dropped transaction.

Table 1-314 Error Source and Handling Table (Sheet 6 of 22)

Event Detector: (If an event is transaction specific, note dependencies in parentheses)	Information Captured: (e.g. Error Status bit, syndrome, bad packet, failing chip indicator, etc. and the CSR register.)	Reporting Mechanism: (e.g. error line, bus interrupt and priority, etc.)	Impact: (e.g. Corrected, Retried, Continues in degraded mode, fatal to chip, Fatal to Domain, Fatal to multiple domains, etc.)
Write Data Parity Error (NCWR, NCBWR, NCWRC targeting Fire containing a parity error or UE error (j_adtype[4] == 1). This error is specific to the data cycles of the PIO write only.)	<p>A maskable error is logged in JBC Error Status Clear Register bit 13/45 (primary/secondary).</p> <p>The following fields are logged in the JBusint In Transaction Error Log Register:</p> <ul style="list-style-type: none"> * Address of the write txn. * j_adtype[5:0] (source device's transid). * Quad word offset - The data beat containing the error. <p>The following fields are logged in the JBusint In Transaction Error Log Register 2:</p> <ul style="list-style-type: none"> * Last 8 cycles worth of JBus arb winners * j_req line state when error occurred * j_pack lines when error occurred. 	A maskable interrupt is generated.	<ul style="list-style-type: none"> * The PIO write is dropped by the j_jbusint block * All other processing of JBus txns continues as normal *PLEASE NOTE: If the write was a write to the MMU flush register, NO FLUSH WILL HAPPEN as the write will be dropped.
Read Data Parity Error (RDS, RDD, RDSA, RDO issued by Fire. This error is specific to the read return data cycles of these txns)	<p>A maskable error is logged in JBC Error Status Clear Register bit 12/44 (primary/secondary).</p> <p>The following fields are logged in the JBusint In Transaction Error Log Register:</p> <ul style="list-style-type: none"> * Address of the original read txn. * j_adtype[5:0] (Fire's transid) captured during the first cycle of the data return. * Quad word offset - The data beat containing the error. <p>The following fields are logged in the JBusint In Transaction Error Log Register 2:</p> <ul style="list-style-type: none"> * Last 8 cycles worth of JBus arb winners * j_req line state when error occurred * j_pack lines when error occurred. 	A maskable interrupt is generated.	<ul style="list-style-type: none"> * For RDS, RDD, and RDSA, the read transaction is completed in error to the DMC core with 4 data beats, some of the data beats have bad parity. * For an RDO, processing will continue as normal and a write completion will eventually be sent to the DMC core; Fire will retain ownership of the cacheline so the WRB will occur normally but with a UE error indicated on the bad data.

Table 1-314 Error Source and Handling Table (Sheet 7 of 22)

Event Detector: (If an event is transaction specific, note dependencies in parentheses)	Information Captured: (e.g. Error Status bit, syndrome, bad packet, failing chip indicator, etc. and the CSR register.)	Reporting Mechanism: (e.g. error line, bus interrupt and priority, etc.)	Impact: (e.g. Corrected, Retried, Continues in degraded mode, fatal to chip, Fatal to Domain, Fatal to multiple domains, etc.)
Illegal NCWR Bytemask (NCWR targeting Fire)	<p>A maskable error is logged in JBC Error Status Clear Register bit 11/ 43 (primary/secondary).</p> <p>The following fields are logged in the JBusint In Transaction Error Log Register:</p> <ul style="list-style-type: none"> * Address of the original PIO txn. * j_adtype[5:0] (source device's transid). <p>The following fields are logged in the JBusint In Transaction Error Log Register 2:</p> <ul style="list-style-type: none"> * Last 8 cycles worth of JBus arb winners * j_req line state when error occurred * j_pack lines when error occurred. 	A maskable interrupt is generated.	The NCWR transaction is dropped by the j_jbusint block.
Illegal NCRD Bytemask (NCRD targeting Fire)	<p>A maskable error is logged in JBC Error Status Clear Register bit 10/ 42 (primary/secondary).</p> <p>The following fields are logged in the JBusint In Transaction Error Log Register:</p> <ul style="list-style-type: none"> * Address of the original PIO txn. * j_adtype[5:0] (source device's transid). <p>The following fields are logged in the JBusint In Transaction Error Log Register 2:</p> <ul style="list-style-type: none"> * Last 8 cycles worth of JBus arb winners * j_req line state when error occurred * j_pack lines when error occurred. 	<p>A maskable interrupt is generated.</p> <p>The j_dmcint block will generate a JBus read response error packet that indicates a bus error.</p>	The NCRD transaction is sent to the j_dmcint block with an error indicated. ¹

Table 1-314 Error Source and Handling Table (Sheet 8 of 22)

Event Detector: (If an event is transaction specific, note dependencies in parentheses)	Information Captured: (e.g. Error Status bit, syndrome, bad packet, failing chip indicator, etc. and the CSR register.)	Reporting Mechanism: (e.g. error line, bus interrupt and priority, etc.)	Impact: (e.g. Corrected, Retried, Continues in degraded mode, fatal to chip, Fatal to Domain, Fatal to multiple domains, etc.)
Invalid JBus Transaction (any reserved JBus txn encoding will trigger this error)	<p>A maskable error is logged in JBC Error Status Clear Register bit 9/41 (primary/secondary).</p> <p>The following fields are logged in the JBusint In Transaction Error Log Register:</p> <ul style="list-style-type: none"> * Address of the txn. * j_adtype[5:0] (source device's transid). <p>The following fields are logged in the JBusint In Transaction Error Log Register 2:</p> <ul style="list-style-type: none"> * Last 8 cycles worth of JBus arb winners * j_req line state when error occurred * j_pack lines when error occurred. 	A maskable interrupt is generated.	<ul style="list-style-type: none"> * Transaction is dropped by the j_jbusint block * All other processing of JBus txns continues as normal
Unmapped PIO Read Error (PIO read to an address not within Fire's range including NCBRD to 8 MB space, NC{B}RD to Fire's cacheable space, and any NC{B}RD targeting Fire that doesn't map to any valid Fire space. Detected by either the j_jbusint or j_dmcint block during address decode)	<p>A maskable error is logged in the JBC Error Status Clear Register bit 28/60 (primary/secondary).</p> <p>The following info is logged in the Dmcint ODCD Error Log Register:</p> <ul style="list-style-type: none"> * address of the PIO read txn * agentid/transid of txn * trans[4:0] txn type encoding 	<p>A maskable interrupt is generated.</p> <p>The j_dmcint block will generate a JBus read response error packet that indicates an unmapped error.</p>	* PIO reads w/ errors are forwarded by the j_dmcint block to the j_csr block in order for a Read Return Error to be issued on JBus. Processing of other JBus commands continues normally. ¹
Unmapped PIO Write Error (PIO write to an address not within Fire's range. Detected by j_dmcint block during address decode)	<p>A maskable error is logged in the JBC Error Status Clear Register bit 8/40 (primary/secondary).</p> <p>The following info is logged in the Dmcint ODCD Error Log Register:</p> <ul style="list-style-type: none"> * address of the PIO write txn * agentid/transid of txn * trans[4:0] txn type encoding 	A maskable interrupt is generated.	* PIO writes are dropped by the j_dmcint block.

Table 1-314 Error Source and Handling Table (Sheet 9 of 22)

Event Detector: (If an event is transaction specific, note dependencies in parentheses)	Information Captured: (e.g. Error Status bit, syndrome, bad packet, failing chip indicator, etc. and the CSR register.)	Reporting Mechanism: (e.g. error line, bus interrupt and priority, etc.)	Impact: (e.g. Corrected, Retried, Continues in degraded mode, fatal to chip, Fatal to Domain, Fatal to multiple domains, etc.)
PIO Data Parity Error (Valid PIO writes to Fire. Detected by j_dmcint block during PIO data RAM read of the data phase of the PIO write)	A maskable error is logged in the JBC Error Status Clear Register bit 7/39 (primary/secondary). The following info is logged in the Dmcint ODCD Error Log Register: * address of the PIO write txn * agentid/transid of txn * trans[4:0] txn type encoding	A maskable interrupt is generated.	The transaction is dropped by the j_dmcint block if it was destined for the j_csr block. It is forwarded to DMC in error.
PIO Command Parity Error (Valid PIO reads and writes to Fire. Detected by j_dmcint block during PIO data RAM read of the address phase of the PIO read/write)	A maskable error is logged in the JBC Error Status Clear Register bit 6/38 (primary/secondary). The following info is logged in the Fatal Error Log Register 1: * address of the PIO read/write txn * agentid/transid of txn * trans[4:0] txn type encoding	A maskable interrupt is generated. A maskable system reset is generated.	* This is a FATAL error and will result in a system reset (4 DOK_ONs) unless this behavior is masked by system software. Fire will not recover from this error, regardless of whether a system reset occurs or not. * The PIO read/write transaction is dropped by the j_dmcint block.
Invalid PIO write to PCIE Cfg/IO, CSR, EBUS or I2C space (PIO write that violates the size access rules for a particular region)	A maskable error is logged in the JBC Error Status Clear Register bit 5/37 (primary/secondary). The following info is logged in the Dmcint ODCD Error Log Register: * address of the PIO write txn * agentid/transid of txn * trans[4:0] txn type encoding	A maskable interrupt is generated.	* PIO writes are dropped by the j_dmcint block.
Invalid PIO read to PCIE Cfg/IO, CSR, EBUS or I2C space (PIO read that violates the size access rules for a particular region)	A maskable error is logged in the JBC Error Status Clear Register bit 27/59 (primary/secondary). The following info is logged in the Dmcint ODCD Error Log Register: * address of the PIO read txn * agentid/transid of txn * trans[4:0] txn type encoding	A maskable interrupt is generated. The j_dmcint block will generate a JBus read response error packet that indicates a bus error.	* PIO reads w/ errors are forwarded by the j_dmcint block to the j_csr block in order for a Read Return Error to be issued on JBus. Processing of other JBus commands continues normally.

Table 1-314 Error Source and Handling Table (Sheet 10 of 22)

Event Detector: (If an event is transaction specific, note dependencies in parentheses)	Information Captured: (e.g. Error Status bit, syndrome, bad packet, failing chip indicator, etc. and the CSR register.)	Reporting Mechanism: (e.g. error line, bus interrupt and priority, etc.)	Impact: (e.g. Corrected, Retried, Continues in degraded mode, fatal to chip, Fatal to Domain, Fatal to multiple domains, etc.)
Unexpected Read Response (Read return to a RDD, RDS or RDSA that Fire did not issue. Detected by j_dmcint block during tag look-up on IDC scoreboard)	A maskable error is logged in the JBC Error Status Clear Register bit 4/36 (primary/secondary). The following info is logged in the Dmcint IDC Error Log Register: *agentid/transid of unexpected read return transaction	A maskable interrupt is generated.	Transaction is dropped by the j_dmcint block.
Unsolicited INT ACK/NACK (Int Ack/Nack to an Int Req that Fire did not issue. Detected by j_dmcint block during tag look-up on IDC scoreboard)	A maskable error is logged in the JBC Error Status Clear Register bit 3/35 (primary/secondary). The following info is logged in the Dmcint IDC Error Log Register: *targID/srcID of unsolicited int ack/nack transaction	A maskable interrupt is generated.	Transaction is dropped by the j_dmcint block.
DMA Write Ack Timeout (Detected by j_dmcint block when a transaction tag for a DMA write issued by a DMC core resides in the IDC scoreboard for a programmed amount of time)	A maskable error is logged in the JBC Error Status Clear Register bit 2/34 (primary/secondary). Fatal Error Log Register 1 will be loaded with 0's on this error.	A maskable interrupt is generated. A maskable system reset is generated.	* This is a FATAL error and will result in a system reset (4 DOK_ONs) unless this behavior is masked by system software. Fire will not recover from this error, regardless of whether a system reset occurs or not. * Completions are not generated for issue to the DMC cores.
Interrupt Response Timeout (Detected by j_dmcint block when a transaction tag for an INT Req issued by a DMC core resides in the IDC scoreboard for a programmed amount of time)	A maskable error is logged in the JBC Error Status Clear Register bit 1/33 (primary/secondary). The following info is logged in the Fatal Error Log Register 1: * dmc_tag issued by a DMC core with the INT transaction * targID/srcID of INT transaction	A maskable interrupt is generated. A maskable system reset is generated.	* This is a FATAL error and will result in a system reset (4 DOK_ONs) unless this behavior is masked by system software. Fire will not recover from this error, regardless of whether a system reset occurs or not. * Completions are not generated for issue to the DMC cores.
DMA Read Response Timeout (Detected by j_dmcint block when a transaction tag for a DMA read issued by a DMC core resides in the IDC scoreboard for a programmed amount of time)	A maskable error is logged in the JBC Error Status Clear Register bit 0/32 (primary/secondary). The following info is logged in the Fatal Error Log Register 1: * dmc_tag issued by a DMC core with the DMA Rd transaction * agentid/transid of DMA Rd transaction	A maskable interrupt is generated. A maskable system reset is generated.	* This is a FATAL error and will result in a system reset (4 DOK_ONs) unless this behavior is masked by system software. Fire will not recover from this error, regardless of whether a system reset occurs or not. * Completions are not generated for issue to the DMC cores.

Table 1-314 Error Source and Handling Table (Sheet 11 of 22)

Event Detector: (If an event is transaction specific, note dependencies in parentheses)	Information Captured: (e.g. Error Status bit, syndrome, bad packet, failing chip indicator, etc. and the CSR register.)	Reporting Mechanism: (e.g. error line, bus interrupt and priority, etc.)	Impact: (e.g. Corrected, Retried, Continues in degraded mode, fatal to chip, Fatal to Domain, Fatal to multiple domains, etc.)
EBUS Ready Time Out Error (Detected by the EBus core in the j_csr block when a Slave Ebus device holds the ready line low for longer than the set time out value.	<p>A maskable error is logged in the JBC Error Status Clear Register bit 26/58 (primary/secondary).</p> <p>The following info is logged in the CSR Error Log Register: The address, byte mask and write bit of the EBUS transaction</p>	A maskable interrupt is generated.	The EBUS transaction is terminated. It may or may not have completed at the slave device. If the transaction was a read whatever data was on the bus at the point of the timeout is returned. This has no effect on any future transactions and processing will continue for other EBUS transactions.
DMC ERRORS			
MSI received but not enabled (DMC IMU - MSI Transaction)	<p>MSI not enabled bit (bit 0) is set in the IMU Error Status Clear register for primary detection. Bit 32 is set for a secondary detection.</p> <p>PCIE transaction information including type, length, req_id, tlp_tag, byte enables and msi data is captured in the IMU RDS Error Log Register.</p>	A maskable interrupt is generated.	The transaction is turned from an MSI to a NULL type in the IMU pipeline and passed up in order to retire the buffer space which has previously been allocated.
MSI Data Parity (DMC IMU - MSI Transaction)	<p>MSI parity error bit (bit 6) is set in the IMU Error Status Clear register for primary detection. Bit 38 is set for a secondary detection.</p> <p>PCIE transaction information including type, length, req_id, tlp_tag, byte enables and msi data is captured in the IMU RDS Error Log Register.</p>	A maskable interrupt is generated.	The transaction is turned from an MSI to a NULL type in the IMU pipeline and passed up in order to retire the buffer space which has previously been allocated.
Malformed MSI (DMC IMU - MSI Transaction)	<p>MSI malformed bit (bit 7) is set in the IMU Error Status Clear register for primary detection. Bit 39 is set for a secondary detection.</p> <p>PCIE transaction information including type, length, req_id, tlp_tag, byte enables and msi data is captured in the IMU RDS Error Log Register.</p>	A maskable interrupt is generated.	The transaction is turned from an MSI to a NULL type in the IMU pipeline and passed up in order to retire the buffer space which has previously been allocated.

Table 1-314 Error Source and Handling Table (Sheet 12 of 22)

Event Detector: (If an event is transaction specific, note dependencies in parentheses)	Information Captured: (e.g. Error Status bit, syndrome, bad packet, failing chip indicator, etc. and the CSR register.)	Reporting Mechanism: (e.g. error line, bus interrupt and priority, etc.)	Impact: (e.g. Corrected, Retried, Continues in degraded mode, fatal to chip, Fatal to Domain, Fatal to multiple domains, etc.)
Correctable Message received but not enabled (DMC IMU - Message Transaction)	Correctable Message not enabled bit (bit 1) is set in the IMU Error Status Clear register for primary detection. Bit 33 is set for a secondary detection. PCIE transaction information including type, length, req_id, tlp_tag, and message code is captured in the IMU RDS Error Log Register.	A maskable interrupt is generated.	The transaction is turned from an Message to a NULL type in the IMU pipeline and passed up in order to retire the buffer space which has previously been allocated.
Non Fatal Message received but not enabled (DMC IMU - Message Transaction)	Non Fatal Message not enabled bit (bit 2) is set in the IMU Error Status Clear register for primary detection. Bit 34 is set for a secondary detection. PCIE transaction information including type, length, req_id, tlp_tag, and message code is captured in the IMU RDS Error Log Register.	A maskable interrupt is generated.	The transaction is turned from an Message to a NULL type in the IMU pipeline and passed up in order to retire the buffer space which has previously been allocated.
Fatal Message received but not enabled (DMC IMU - Message Transaction)	Fatal Message not enabled bit (bit 3) is set in the IMU Error Status Clear register for primary detection. Bit 35 is set for a secondary detection. PCIE transaction information including type, length, req_id, tlp_tag, and message code is captured in the IMU RDS Error Log Register.	A maskable interrupt is generated.	The transaction is turned from an Message to a NULL type in the IMU pipeline and passed up in order to retire the buffer space which has previously been allocated.
PM PME Message received but not enabled (DMC IMU - Message Transaction)	PM PME Message not enabled bit (bit 4) is set in the IMU Error Status Clear register for primary detection. Bit 36 is set for a secondary detection. PCIE transaction information including type, length, req_id, tlp_tag, and message code is captured in the IMU RDS Error Log Register.	A maskable interrupt is generated.	The transaction is turned from an Message to a NULL type in the IMU pipeline and passed up in order to retire the buffer space which has previously been allocated.

Table 1-314 Error Source and Handling Table (Sheet 13 of 22)

Event Detector: (If an event is transaction specific, note dependencies in parentheses)	Information Captured: (e.g. Error Status bit, syndrome, bad packet, failing chip indicator, etc. and the CSR register.)	Reporting Mechanism: (e.g. error line, bus interrupt and priority, etc.)	Impact: (e.g. Corrected, Retried, Continues in degraded mode, fatal to chip, Fatal to Domain, Fatal to multiple domains, etc.)
PME to ACK Message received but not enabled (DMC IMU - Message Transaction)	PME to ACK Message not enabled bit (bit 5) is set in the IMU Error Status Clear register for primary detection. Bit 37 is set for a secondary detection. PCIE transaction information including type, length, req_id, tlp_tag, and message code is captured in the IMU RDS Error Log Register.	A maskable interrupt is generated.	The transaction is turned from an Message to a NULL type in the IMU pipeline and passed up in order to retire the buffer space which has previously been allocated.
EQ Write to non enabled EQ (DMC IMU - Message/MSI Transaction)	EQ not enabled bit (bit 8) is set in the IMU Error Status Clear register for primary detection. Bit 40 is set for a secondary detection. PCIE transaction information including type, length, req_id, tlp_tag, and byte enables are captured in the IMU SCS Error Log Register.	A maskable interrupt is generated.	The transaction is turned from an Message/MSI to a NULL type in the IMU pipeline and passed up in order to retire the buffer space which has previously been allocated.
EQ Over Flow Error (DMC IMU - Message/MSI Transaction)	EQ Overflow bit (bit 9) is set in the IMU Error Status Clear register for primary detection. Bit 41 is set for a secondary detection. No other transaction information is captured.	A maskable interrupt is generated.	The transaction is turned from an Message/MSI to a NULL type in the IMU pipeline and passed up in order to retire the buffer space which has previously been allocated. Also the EQ state will transition from ACTIVE to ERROR.
MMU Bypass Mode Error (any translation request using 64-bit addresses)	MMU Error Status Clear Register bit BYP_ERR is set. Virtual address is logged in the MMU Translation Fault Address Register. ID and Type are logged in the MMU Translation Fault Status Register.	A maskable interrupt is generated.	Write transaction is terminated. Read transaction is completed with Completer Abort Status.
MMU Bypass Mode Out-of-Range Error (any translation request using 64-bit addresses)	MMU Error Status Clear Register bit BYP_OOR is set. Virtual address is logged in the MMU Translation Fault Address Register. ID and Type are logged in the MMU Translation Fault Status Register.	A maskable interrupt is generated.	Write transaction is terminated. Read transaction is completed with Completer Abort Status.

Table 1-314 Error Source and Handling Table (Sheet 14 of 22)

Event Detector: (If an event is transaction specific, note dependencies in parentheses)	Information Captured: (e.g. Error Status bit, syndrome, bad packet, failing chip indicator, etc. and the CSR register.)	Reporting Mechanism: (e.g. error line, bus interrupt and priority, etc.)	Impact: (e.g. Corrected, Retried, Continues in degraded mode, fatal to chip, Fatal to Domain, Fatal to multiple domains, etc.)
MMU Translation Mode Error (any translation request using 32-bit addresses)	MMU Error Status Clear Register bit TRN_ERR is set. Virtual address is logged in the MMU Translation Fault Address Register. ID and Type are logged in the MMU Translation Fault Status Register.	A maskable interrupt is generated.	Write transaction is terminated. Read transaction is completed with Completer Abort Status.
MMU Translation Mode Out-of-Range Error (any translation request using 32-bit addresses)	MMU Error Status Clear Register bit TRN_OOR is set. Virtual address is logged in the MMU Translation Fault Address Register. ID and Type are logged in the MMU Translation Fault Status Register.	A maskable interrupt is generated.	Write transaction is terminated. Read transaction is completed with Completer Abort Status.
MMU Translation Table Entry Invalid Error (any translation request using 32-bit addresses)	MMU Error Status Clear Register bit TTE_INV is set. Virtual address is logged in the MMU Translation Fault Address Register. ID and Type are logged in the MMU Translation Fault Status Register.	A maskable interrupt is generated.	Write transaction is terminated. Read transaction is completed with Completer Abort Status.
MMU Translation Table Entry Protection Error (any translation request using 32-bit addresses)	MMU Error Status Clear Register bit TTE_PRT is set. Virtual address is logged in the MMU Translation Fault Address Register. ID and Type are logged in the MMU Translation Fault Status Register.	A maskable interrupt is generated.	Write transaction is terminated. Read transaction is completed with Completer Abort Status.
MMU Translation Table Cache Data Parity Error (any translation request using 32-bit addresses)	MMU Error Status Clear Register bit TTC_DPE is set. Virtual address is logged in the MMU Translation Fault Address Register. ID, Type, and RAM address are logged in the MMU Translation Fault Status Register.	A maskable interrupt is generated.	Write transaction is terminated. Read transaction is completed with Completer Abort Status.
MMU Translation Table Cache Access Error (any CSR access to the cache while the cache is enabled)	MMU Error Status Clear Register bit TTC_CAE set.	A maskable interrupt is generated.	Write fails, read returns error status.

Table 1-314 Error Source and Handling Table (Sheet 15 of 22)

Event Detector: (If an event is transaction specific, note dependencies in parentheses)	Information Captured: (e.g. Error Status bit, syndrome, bad packet, failing chip indicator, etc. and the CSR register.)	Reporting Mechanism: (e.g. error line, bus interrupt and priority, etc.)	Impact: (e.g. Corrected, Retried, Continues in degraded mode, fatal to chip, Fatal to Domain, Fatal to multiple domains, etc.)
MMU Tablewalk Disabled Miss Error (any miss in the cache of translation request)	MMU Error Status Clear Register bit TBW_DME is set. Virtual address is logged in the MMU Translation Fault Address Register. ID and Type are logged in the MMU Translation Fault Status Register.	A maskable interrupt is generated.	Write transaction is terminated. Read transaction is completed with Completer Abort Status.
MMU Tablewalk Unexpected Data Error (any miss in the cache of translation request)	MMU Error Status Clear Register bit TBW_UDE is set. Virtual address is logged in the MMU Translation Fault Address Register. ID and Type are logged in the MMU Translation Fault Status Register.	A maskable interrupt is generated.	Write transaction is terminated. Read transaction is completed with Completer Abort Status.
MMU Tablewalk Error (any translation request using 32-bit addresses)	MMU Error Status Clear Register bit TBW_ERR set. Virtual address is logged in the MMU Translation Fault Address Register. ID and Type are logged in the MMU Translation Fault Status Register.	A maskable interrupt is generated.	Write transaction is terminated. Read transaction is completed with Completer Abort Status.
MMU Tablewalk Data Parity Error (any translation request using 32-bit addresses)	MMU Error Status Clear Register bit TBW_DPE is set. Virtual address is logged in the MMU Translation Fault Address Register. ID and Type are logged in the MMU Translation Fault Status Register.	A maskable interrupt is generated.	Write transaction is terminated. Read transaction is completed with Completer Abort Status.
PEC ERRORS			
Ingress header parity error (from IHB to ILU)	ILU Error Status Clear Register bit (IHB_PE) is set.	A maskable interrupt is generated.	Header is dropped. Drain state - soft reset is needed to take Fire out of the Drain State caused by this error. Fatal error
Ingress IO Rd	TLU Uncorrectable Error Status Clear Register bit (UR) is set. Request header is logged in TLU Receive Uncorrectable Error Header1 and Header2 Log Registers.	A maskable interrupt is generated.	Completed with Unsupported Request Status.

Table 1-314 Error Source and Handling Table (Sheet 16 of 22)

Event Detector: (If an event is transaction specific, note dependencies in parentheses)	Information Captured: (e.g. Error Status bit, syndrome, bad packet, failing chip indicator, etc. and the CSR register.)	Reporting Mechanism: (e.g. error line, bus interrupt and priority, etc.)	Impact: (e.g. Corrected, Retried, Continues in degraded mode, fatal to chip, Fatal to Domain, Fatal to multiple domains, etc.)
Ingress IO Wr (for both cases of data poisoned and not poisoned)	TLU Uncorrectable Error Status Clear Register bit (UR) is set. Request header is logged in TLU Receive Uncorrectable Error Header1 and Header2 Log Registers.	A maskable interrupt is generated.	Completed with Unsupported Request Status.
Ingress Cfg Rd	TLU Uncorrectable Error Status Clear Register bit (UR) is set. Request header is logged in TLU Receive Uncorrectable Error Header1 and Header2 Log Registers.	A maskable interrupt is generated.	Completed with Unsupported Request Status.
Ingress Cfg Wr (for both cases of data poisoned and not poisoned)	TLU Uncorrectable Error Status Clear Register bit (UR) is set. Request header is logged in TLU Receive Uncorrectable Error Header1 and Header2 Log Registers.	A maskable interrupt is generated.	Completed with Unsupported Request Status.
Ingress Mem Rd Lock	TLU Uncorrectable Error Status Clear Register bit (UR) is set. Request header is logged in TLU Receive Uncorrectable Error Header1 and Header2 Log Registers.	A maskable interrupt is generated.	Completed with Unsupported Request Status.
Ingress unexpected Completion (mismatches in tag or req ID; ingress CplLk; ingress CplDLk)	TLU Uncorrectable Error Status Clear Register bit (UC) is set. Completion header is logged in Receive Uncorrectable Error Header1 and Header2 Log Registers.	A maskable interrupt is generated.	Packet is dropped. Software determines impact.
Ingress completion header error (mismatches in byte cnt (chk can be disabled), addr (chk can be disabled), length, TC (chk can be disabled) or Attr (chk can be disabled); CplID with unsuccessful status; config-retry status for non-config reqs (chk can be disabled), CplID to PIO wr reqs; Successful Cpl to PIO rd reqs)	TLU Other Event Status Clear Register bit (MFC) is set. Completion header is logged in Receive Other Event Header1 and Header2 Log Registers. Original request header is logged in TLU Transmit Other Event Header1 and Header2 Log Registers.	A maskable interrupt is generated.	Packet is dropped. An associated completion with time out error status is generated by Fire. Software determines impact.

Table 1-314 Error Source and Handling Table (Sheet 17 of 22)

Event Detector: (If an event is transaction specific, note dependencies in parentheses)	Information Captured: (e.g. Error Status bit, syndrome, bad packet, failing chip indicator, etc. and the CSR register.)	Reporting Mechanism: (e.g. error line, bus interrupt and priority, etc.)	Impact: (e.g. Corrected, Retried, Continues in degraded mode, fatal to chip, Fatal to Domain, Fatal to multiple domains, etc.)
Ingress MsgD (for both cases of data poisoned and not poisoned) (exclude "Vendor Defined Type-1" messages, which are silently dropped)	TLU Uncorrectable Error Status Clear Register bit (UR) is set. Header is logged in TLU Receive Uncorrectable Error Header1 and Header2 Log Registers.	A maskable interrupt is generated.	Packet is dropped. Credits are recovered. Software determines impact.
Completion timeout	TLU Uncorrectable Error Status Clear Register bit (CTO) is set. Original request header is logged in TLU Transmit Uncorrectable Error Header1 and Header2 Log Registers. TLU Other Event Status Clear Register bit (CTO) is set. Original request header is logged in TLU Transmit Other Event Header1 and Header2 Log Registers.	Two maskable interrupts are generated.	Completion is generated with timeout error status. Software determines impact.
Ingress MWr request (posted) with poisoned data	TLU Uncorrectable Error Status Clear Register bit (PP) is set. Request header is logged in TLU Receive Uncorrectable Error Header1 and Header2 Log Registers.	A maskable interrupt is generated.	Packet is dropped. Credits are recovered. Software determines impact.
Ingress CplID with poisoned data	TLU Uncorrectable Error Status Clear Register bit (PP) is set. Completion header is logged in Receive Uncorrectable Error Header1 and Header2 Log Registers.	A maskable interrupt is generated.	Packet is dropped. Software determines impact.
Ingress malformed packet	TLU Uncorrectable Error Status Clear Register bit (MFP) is set. Header is logged in TLU Receive Uncorrectable Error Header1 and Header2 Log Registers.	A maskable interrupt is generated.	Packet is dropped. Credits are not recovered. Software determines impact.
Potential Link width change (when LTSSM Recovery leads to (re)configuration.)	TLU Other Event Status Clear Register bit (LWC) is set.	A maskable interrupt is generated.	All the timer threshold registers in PEC should be reconfigured if the link width does change due to the (re)configuration.
No forward progress (When two consecutive retrains occur with no valid TLP received between them.)	TLU Other Event Status Clear Register bit (NFP) is set.	A maskable interrupt is generated.	An indication that the remote device might be gummed up and causes a possible livelock in PCI Express. Software determines impact.

Table 1-314 Error Source and Handling Table (Sheet 18 of 22)

Event Detector: (If an event is transaction specific, note dependencies in parentheses)	Information Captured: (e.g. Error Status bit, syndrome, bad packet, failing chip indicator, etc. and the CSR register.)	Reporting Mechanism: (e.g. error line, bus interrupt and priority, etc.)	Impact: (e.g. Corrected, Retried, Continues in degraded mode, fatal to chip, Fatal to Domain, Fatal to multiple domains, etc.)
Ingress receiver overflow (credit violation)	TLU Uncorrectable Error Status Clear Register bit (ROF) is set. Header is logged in TLU Receive Uncorrectable Error Header1 and Header2 Log Registers.	A maskable interrupt is generated.	Packet is dropped. Credits are not recovered. Software determines impact.
Flow control protocol error (This happens when there are negative egress credits.)	TLU Uncorrectable Error Status Clear Register bit (FCP) is set.	A maskable interrupt is generated.	This is caused by remote device's misbehavior. Software determines impact.
Data link protocol error (This occurs when the remote device sends an ACK/NAK with an invalid sequence number.)	TLU Uncorrectable Error Status Clear Register bit (DLP) is set. LPU Link Layer Interrupt and Status Register bit (INT_DLNK_PES) is set.	A maskable interrupt is generated.	This is caused by remote device's misbehavior. Software determines impact.
Training error (It occurs when LTSSM fails in Configuration state)	TLU Uncorrectable Error Status Clear Register bit (TE) is set. LPU Phy Layer Interrupt Register bit (INT_TRN_ERR) is set.	A maskable interrupt is generated.	Link is unavailable. Fatal error
Replay timer timeout	TLU Correctable Error Status Clear Register bit (RTO) is set. Link Layer Error Register bit (INT_RPLAY_TMR_TO) is set.	A maskable interrupt is generated.	Corrected by retry.
REPLAY_NUM rollover	TLU Correctable Error Status Clear Register bit (RNR) is set. Link Layer Error Register bit (INT_RPLAY_NUM_RO) is set.	A maskable interrupt is generated.	Corrected by retraining the link through LTSSM Recovery state (link is not down).
Bad DLLP error	TLU Correctable Error Status Clear Register bit (BDP) is set. Link Layer Interrupt and Status Register bit (INT_BAD_DLLP) is set.	A maskable interrupt is generated.	
Bad TLP error	TLU Correctable Error Status Clear Register bit (BTP) is set. Link Layer Error Register bit (INT_BAD_TLP) is set. LPU Phy Layer Error Interrupt Register bit (Receiver Error) is set.	A maskable interrupt is generated.	
Receiver error	TLU Correctable Error Status Clear Register bit (RE) is set. LPU Link Layer Interrupt and Status Register bit (INT_DLLP_RCV_ERR or INT_TLP_RCV_ERR) is set. LPU Phy Layer Error Interrupt Register bit is set.	A maskable interrupt is generated.	

Table 1-314 Error Source and Handling Table (Sheet 19 of 22)

Event Detector: (If an event is transaction specific, note dependencies in parentheses)	Information Captured: (e.g. Error Status bit, syndrome, bad packet, failing chip indicator, etc. and the CSR register.)	Reporting Mechanism: (e.g. error line, bus interrupt and priority, etc.)	Impact: (e.g. Corrected, Retried, Continues in degraded mode, fatal to chip, Fatal to Domain, Fatal to multiple domains, etc.)
Memory read capture	TLU Other Event Status Clear Register bit (MRC) is set. Request header is logged in Receive Other Event Header1 and Header2 Log Registers.	A maskable interrupt is generated.	Diagnostic Read request is removed from the pipeline and requires software generated completion. No flow control credit is collected.
PIO write request completion with unsuccessful status	TLU Other Event Status Clear Register bit (WUC) is set. Completion header is logged in Receive Other Event Header1 and Header2 Log Registers. Original request header is logged in TLU Transmit Other Event Header1 and Header2 Log Registers.	A maskable interrupt is generated.	Software determines impact.
PIO read request completion with unsuccessful status	TLU Other Event Status Clear Register bit (RUC) is set. Completion header is logged in Receive Other Event Header1 and Header2 Log Registers. Original request header is logged in TLU Transmit Other Event Header1 and Header2 Log Registers.	A maskable interrupt is generated.	Software determines impact.
Completion with configuration retry status	TLU Other Event Status Clear Register bit (CRS) is set. Completion header is logged in Receive Other Event Header1 and Header2 Log Registers. Original request header is logged in TLU Transmit Other Event Header1 and Header2 Log Registers.	A maskable interrupt is generated.	Completion packet is dropped and a replacement completion with a timeout completion status is generated. Software determines impact.
Ingress interface parity error	TLU Other Event Status Clear Register bit (IIP) is set.	A maskable interrupt is generated.	Packet is forwarded with bad parity. If the parity error is associated with header, it causes a fatal PEC Ingress header parity error. If the parity error is associated with payload, it causes JBC Merge Buffer Parity Error (wr_buf).

Table 1-314 Error Source and Handling Table (Sheet 20 of 22)

Event Detector: (If an event is transaction specific, note dependencies in parentheses)	Information Captured: (e.g. Error Status bit, syndrome, bad packet, failing chip indicator, etc. and the CSR register.)	Reporting Mechanism: (e.g. error line, bus interrupt and priority, etc.)	Impact: (e.g. Corrected, Retried, Continues in degraded mode, fatal to chip, Fatal to Domain, Fatal to multiple domains, etc.)
Egress data parity error (from EDB)	TLU Other Event Status Clear Register bit (EDP) is set	A maskable interrupt is generated.	Drain state - soft reset is needed to take Fire out of the Drain State caused by this error. Fatal error If it happens during a transaction, the packet is terminated by inverting CRC and inserting EDB frame character. Flow control credits are consumed by Fire, which is a deviation from PCI-E spec.
Egress header parity error (from EHB)	TLU Other Event Status Clear Register bit (EHP) is set.	A maskable interrupt is generated.	Drain state - soft reset is needed to take Fire out of the Drain State caused by this error. Packet is dropped. Fatal error
Link interrupt	TLU Other Event Error Status Clear Register bit (LIN) is set.	A maskable interrupt is generated.	Interrupt from LPU, which can only happen if software enables an LPU interrupt.
Link reset (It occurs when LTSSM enters HOT_RESET state)	TLU Other Event Error Status Clear Register bit (LRS) is set.	A maskable interrupt is generated.	An indication that the remote device wants to reset the link. Software has to make the call.
Link down transition	TLU Other Event Error Status Clear Register bit (LDN) is set.	A maskable interrupt is generated.	Drain state - refer to section of "Re-train a Link After it is Down" for the procedures to get out of the Drain State.
Link up transition	TLU Other Event Error Status Clear Register bit (LUP) is set.	A maskable interrupt is generated.	Status indication of link up
Retry buffer underflow error	TLU Other Event Error Status Clear Register bit (ERU) is set. LPU Link Layer Interrupt and Status Register bit (INT_RTRY_BUF_UDF_ERR) is set.	A maskable interrupt is generated.	Fatal error
Retry buffer overflow error	TLU Other Event Error Status Clear Register bit (ERO) is set. LPU Link Layer Interrupt and Status Register bit (INT_RTRY_BUF_OVF_ERR) is set.	A maskable interrupt is generated.	Fatal error

Table 1-314 Error Source and Handling Table (Sheet 21 of 22)

Event Detector: (If an event is transaction specific, note dependencies in parentheses)	Information Captured: (e.g. Error Status bit, syndrome, bad packet, failing chip indicator, etc. and the CSR register.)	Reporting Mechanism: (e.g. error line, bus interrupt and priority, etc.)	Impact: (e.g. Corrected, Retried, Continues in degraded mode, fatal to chip, Fatal to Domain, Fatal to multiple domains, etc.)
Egress minimum packet error	TLU Other Event Error Status Clear Register bit (EMP) is set. LPU Link Layer Interrupt and Status Register bit (INT_EG_TLP_MIN_ERR) is set.	A maskable interrupt is generated.	Packet is dropped. It's a Fire HW bug if this happens. Software determines impact.
Egress protocol error	TLU Other Event Error Status Clear Register bit (EPE) is set. LPU Link Layer Interrupt and Status Register bit (INT_EG_TRNC_FRM_ERR) is set.	A maskable interrupt is generated.	Fatal error
Egress completer abort (The error bit (CA) will never be set because PEC doesn't check for this error. It's listed here for completeness. However, Fire does complete the incoming PCIe read requests with "complete abort" status and transmit the completions to PCIe port if the incoming PCIe read requests encounter some MMU errors)	TLU Uncorrectable Error Status Clear Register bit (CA) is set.	A maskable interrupt is generated.	None, refer to MMU errors.
Egress retry buffer parity error	TLU Other Event Error Status Clear Register bit (ERP) is set. LPU Link Layer Interrupt and Status Register bit (INT_RTRY_BUF_PE) is set.	A maskable interrupt is generated.	Fatal If it happens during a transaction, the packet is terminated by inverting CRC and inserting EDB frame character. Flow control credits are consumed by Fire, which is a deviation from PCI-E spec.

Table 1-314 Error Source and Handling Table (Sheet 22 of 22)

Event Detector: (If an event is transaction specific, note dependencies in parentheses)	Information Captured: (e.g. Error Status bit, syndrome, bad packet, failing chip indicator, etc. and the CSR register.)	Reporting Mechanism: (e.g. error line, bus interrupt and priority, etc.)	Impact: (e.g. Corrected, Retried, Continues in degraded mode, fatal to chip, Fatal to Domain, Fatal to multiple domains, etc.)
Egress interface parity error	TLU Other Event Error Status Clear Register bit (EIP) is set. LPU Link Layer Interrupt and Status Register bit (INT_EGRESS_PE) is set.	A maskable interrupt is generated.	If it happens during a transaction, the packet is terminated by inverting CRC and inserting EDB frame character. Flow control credits are consumed by Fire, which is a deviation from PCI-E spec.
Ingress unsupported DLLP	LPU Link Layer Interrupt and Status Register bit (INT_UNSPRTED_DLLP) is set. DLLP is logged in LPU Rxlink Unsupported DLLP Received Register.	A maskable interrupt is generated.	
TLP received with inverted CRC and EDB	LPU Link Layer Interrupt and Status Register bit (INT_SRC_ERR_TLP) is set.	A maskable interrupt is generated.	Packet is dropped.

1. Note that it is possible to have both an Illegal NCRD Bytemask error as well as an Unmapped PIO Read Error logged for the same transaction. This is due to the fact that reads with errors are not dropped as a read error packet must be returned and multiple errors may be detected by different blocks (jbusint and dmcint) along the way. This was deemed to not be a problem. The same will not be true for writes since writes are dropped after the first error is detected by the jbusint block.

1.4.14 *Interrupt Processing for Mondos 62 and 63*

1.4.14.1 *Fire Internal JBC Error Mondo 63*

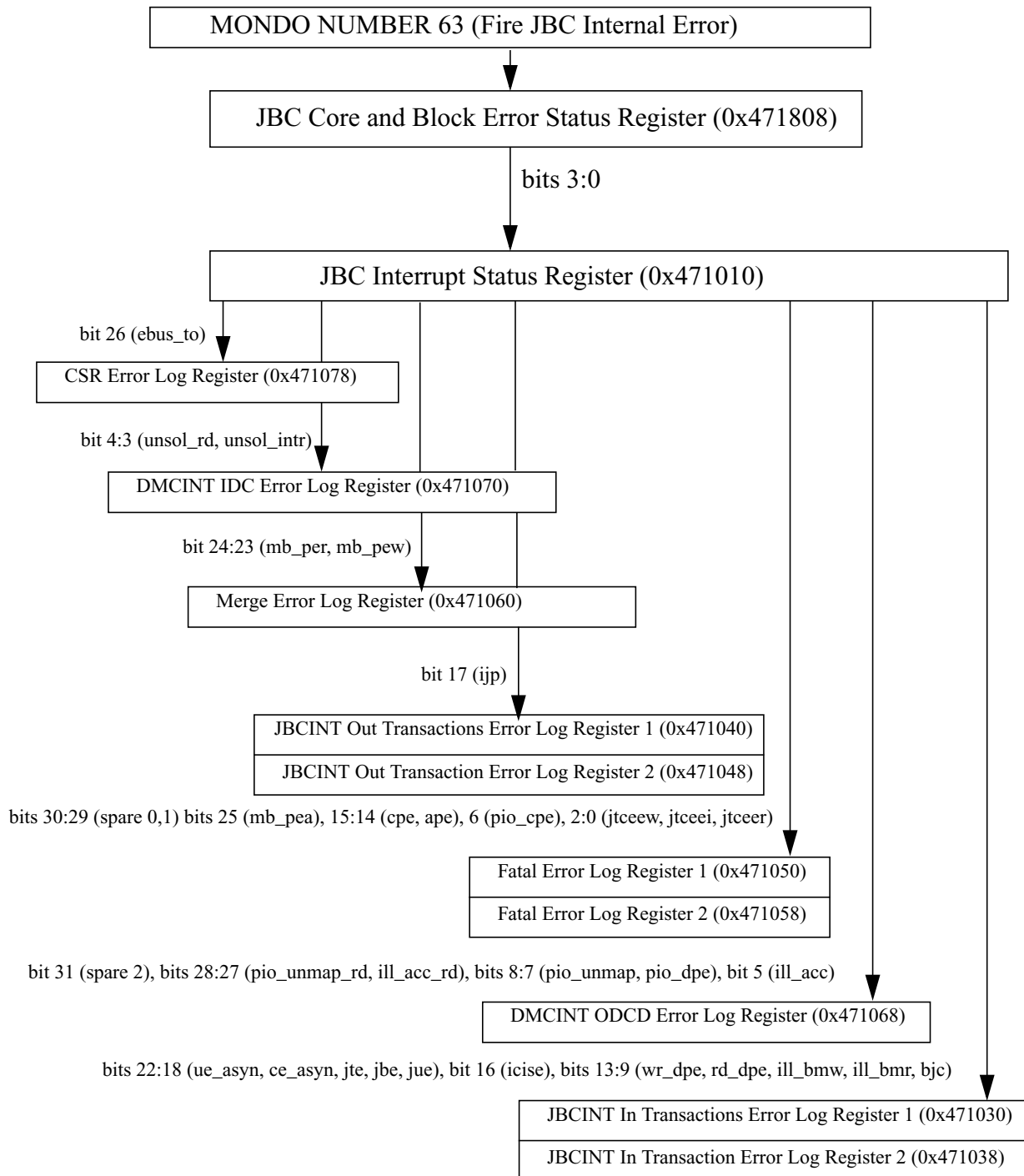
When software receives a Mondo from INO 63, this signifies that an internal Fire error was detected in the JBC Core. Below in the following figures and descriptions are shown the details and the necessary registers that should be read when a mondo 63 is received.

The first register to be read should be the JBC Core and Block Error Status Register (0x471808). This register describes which of one or more than one of the 4 possible block(s) in the JBC Core has an error which needs to be processed.

When this register is read and any of the bits 3:0 are set the next register that should be read is the JBC Interrupt Status Register (0x471010). This register shows all of the possible JBC Errors which could have generated a mondo 63. From these bits further information about the particular error can be obtained by further reading the proper Error logging register. Please see the diagram for details on these registers.

To clear any of the errors detected in the JBC Core, the JBC Error Status Clear Register (0x471018) should be used.

Figure 1-4 Mondo 63 Fire JBC Internal Error Processing Diagram

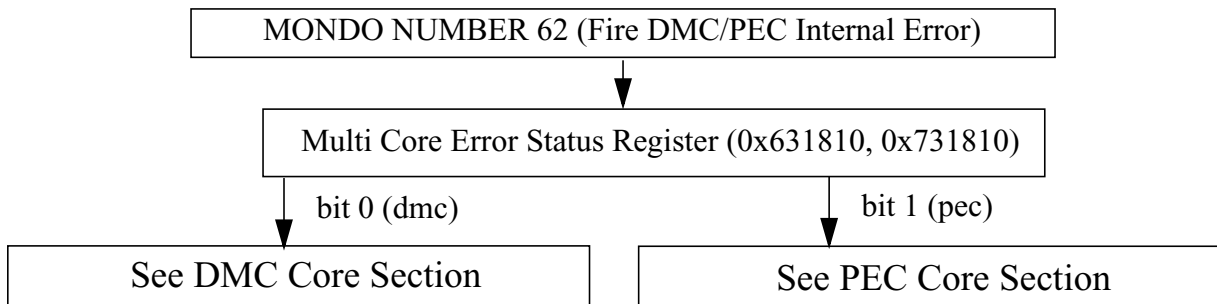


1.4.14.2 Fire Internal DMC & PEC Error or Event Mondo 62

When software receives a Mondo from INO 62, this signifies that an internal Fire error or event was detected in the DMC or PEC Core. Below in the following figures and descriptions are shown the details and the necessary registers that should be read when a mondo 62 is received.

The first register to be read should be the Multi Core Error Status Register (0x631810, 0x731810) This register describes which of one or more than one of the 2 possible cores the Mondo 62 came from. If bit 0 is set in this register the Mondo was caused by the DMC core. If bit 1 is set in this register the Mondo was caused by the PEC Core

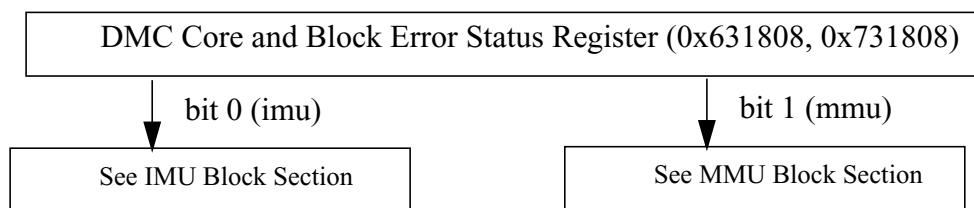
Figure 1-5 Mondo 62 Fire DMC & PEC Internal Error or Event Processing Diagram



1.4.14.2.1 DCM Core

If Mondo 62 was caused by the DMC Core the next register that should be read is the DMC Core and Block Error Status Register (0x631808, 0x731808). This register describes which of one or more than one of the 2 possible block(s) in the DMC Core has an error which needs to be processed. If bit 0 is set in this register the Mondo was caused by the IMU block. If bit 1 is set in this register the Mondo was caused by the MMU block

Figure 1-6 DMC Core Error Processing Diagram

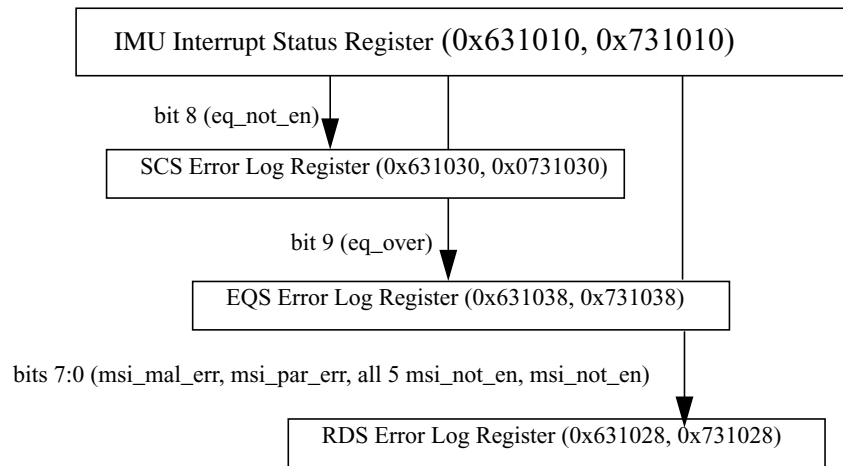


IMU Block

If Mondo 62 was caused by the IMU Block the next register that should be read is IMU Interrupt Status Register (0x631010, 0x731010). This register shows all of the possible IMU Errors which could have generated a mondo 62. From these bits further information about the particular error can be obtained by further reading the proper Error logging register. Please see the diagram for details on these registers.

To clear any of the errors detected in the IMU Block, the IMU Error Status Clear Register (0x631018, 0x731018) should be used.

Figure 1-7 IMU Block Error Processing Diagram

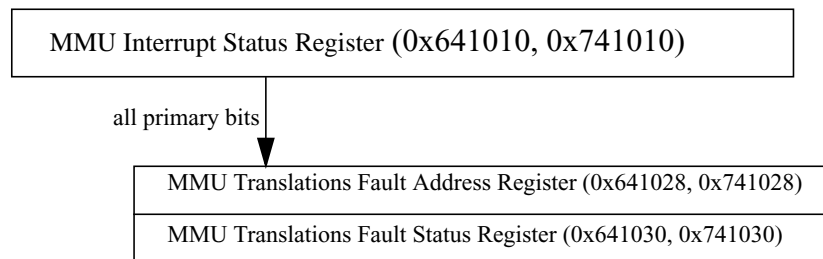


MMU Block

If Mondo 62 was caused by the MMU Block the next register that should be read is MMU Interrupt Status Register (0x641010, 0x741010). This register shows all of the possible MMU Errors which could have generated a mondo 62. From these bits further information about the particular error can be obtained by further reading the proper Error logging register. Please see the diagram for details on these registers.

To clear any of the errors detected in the MMU Block, the IMU Error Status Clear Register (0x641018, 0x741018) should be used.

Figure 1-8 MMU Block Error Processing Diagram

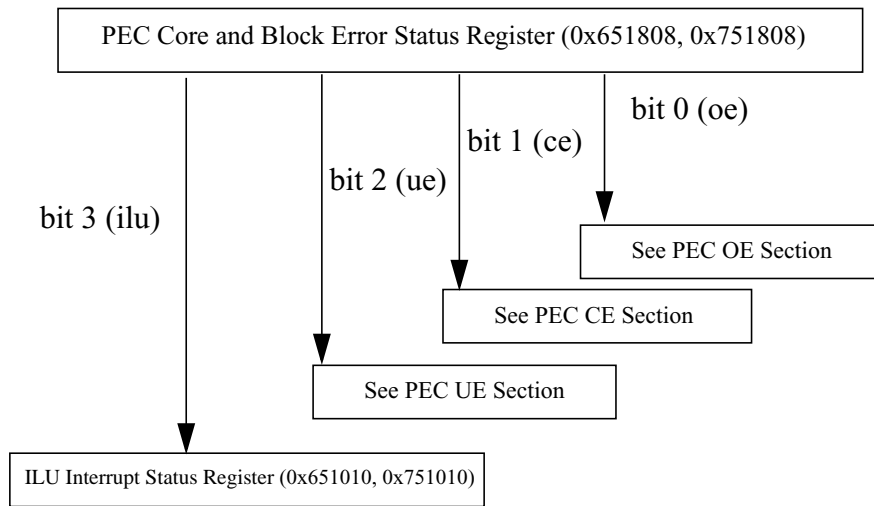


1.4.14.2.2 PEC Core

If Mondo 62 was caused by the PEC Core the next register that should be read is the PEC Core and Block Error Status Register (0x651808, 0x751808). This register describes which one or more than one of the 4 possible block(s) in the PEC Core has an error which needs to be processed. If bit 0 is set in this register the Mondo was caused by the a PEC OE register. If bit 1 is set in this register the Mondo was caused by the PEC CE Register. If bit 2 was set in this register the Mondo was caused by the PEC UE Register. If bit 3 was set in this register the Mondo was caused by the ILU block.

To clear any of the errors detected in the ILU Block, the ILU Error Status Clear Register (0x651018, 0x751018) should be used.

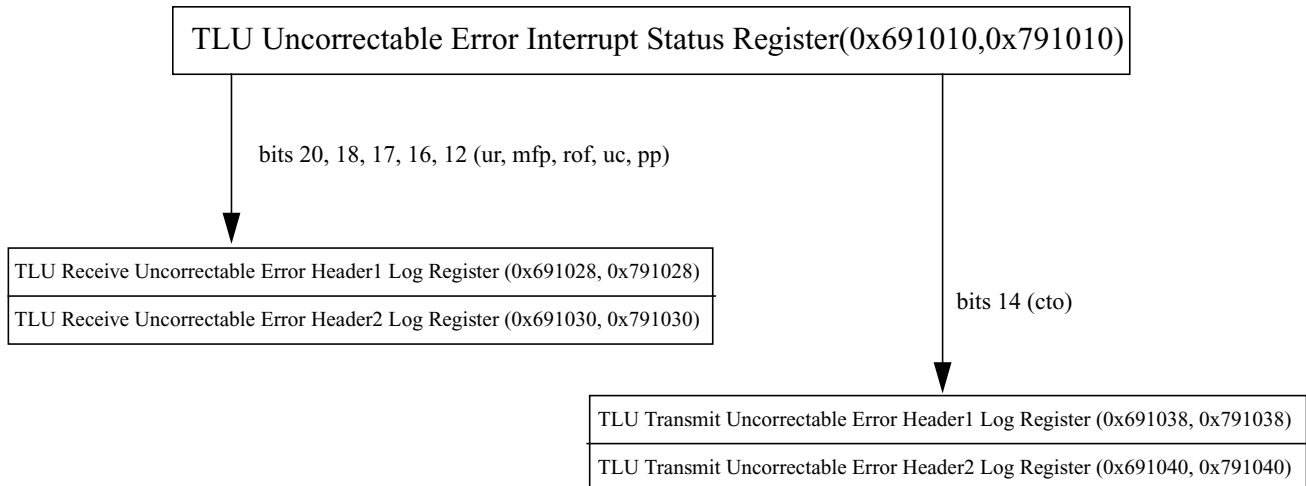
Figure 1-9 PEC Core Error Processing Diagram



PEC UE

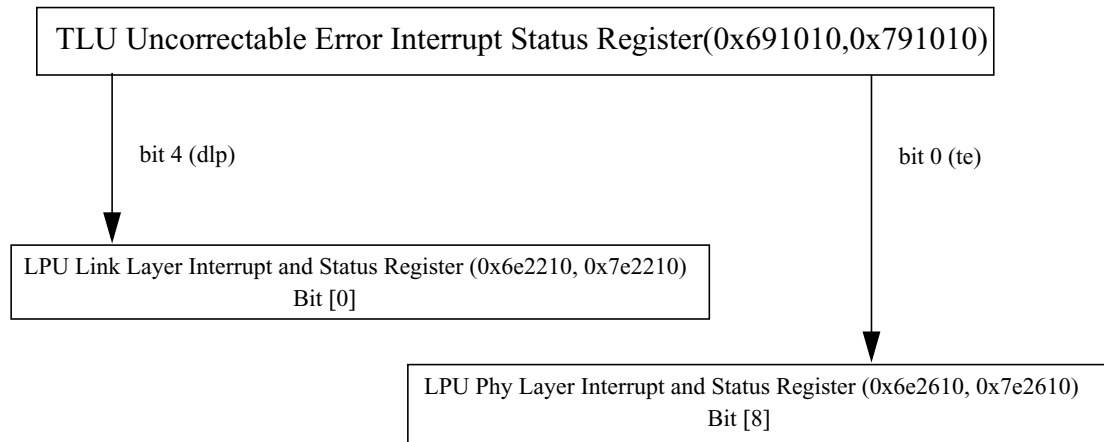
If Mondo 62 was caused by a PEC UE error or event the next register that should be read is TLU Uncorrectable Error Interrupt Status Register(0x691010,0x791010). This register shows all of the possible TLU UE Errors and events which could have generated a mondo 62. From these bits further information about the particular error can be obtained by further reading the proper Error logging register. Please see the diagram for details on these registers. Also note that bits 13 and bits 15 of this register DO NOT store any additional data with the errors and are not show in this diagram.

Figure 1-10 PEC UE Error Processing Diagram



Since this register, can store errors that are detected by the LPU as well as the TLU some of this information may be duplicated in the LPU in a different register. In the following diagram any bits which could store duplicate information are describe along with the register and the bit which would be duplicate.

Figure 1-11 PEC UE Duplicate Error Diagram



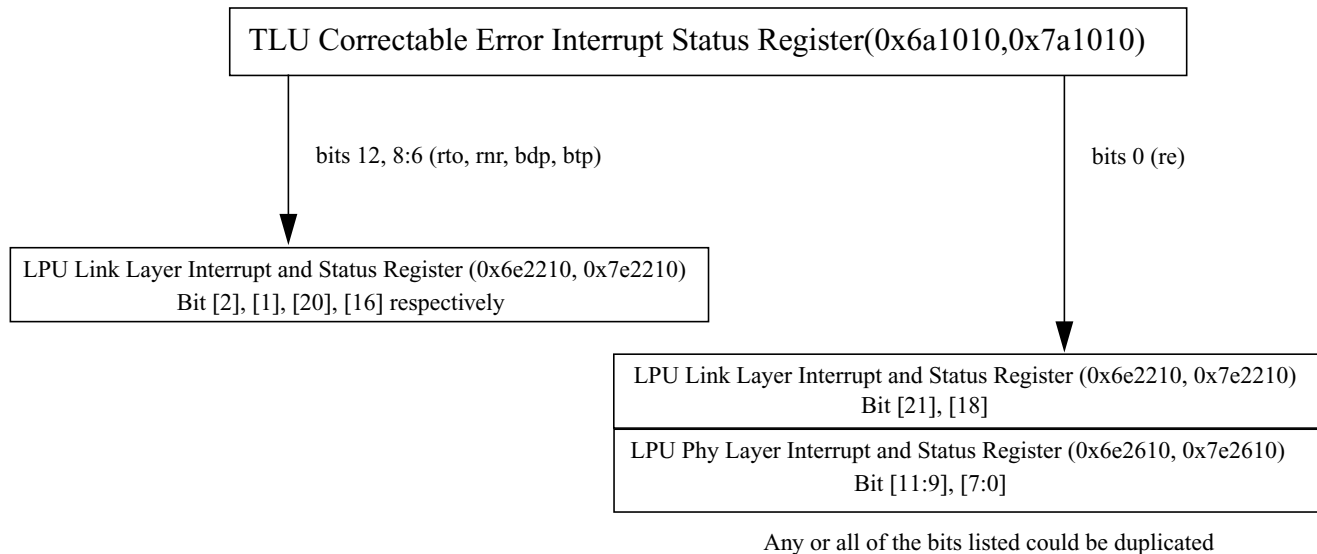
To clear any of the errors detected in the in TLU UE register, the TLU Uncorrectable Error Status Clear Register (0x691018, 0x791018) should be used. To clear the duplicate registers in the LPU the LPU Link Layer Interrupt and Status Register (0x6e2210, 0x7e2210) should be used for the dlp duplicate error (bit 0 of that register) and the LPU Phy Layer Interrupt and Status Register (0x6e2610, 0x7e2610) should be used for the duplicate te error (bit 8 of that register).

PEC CE

If Mondo 62 was caused by a PEC CE error or event the next register that should be read is TLU Correctable Error Interrupt Status Register(0x6a1010,0x7a1010). This register shows all of the possible TLU CE Errors and events which could have generated a mondo 62. From these bits no further information about the particular error is stored

Since this register, can store errors that are detected by the LPU as well as the TLU some of this information may be duplicated in the LPU in a different register. In the following diagram any bits which could store duplicate information are describe along with the register an the bit which would be duplicate.

Figure 1-12 PEC CE Duplicate Error Diagram



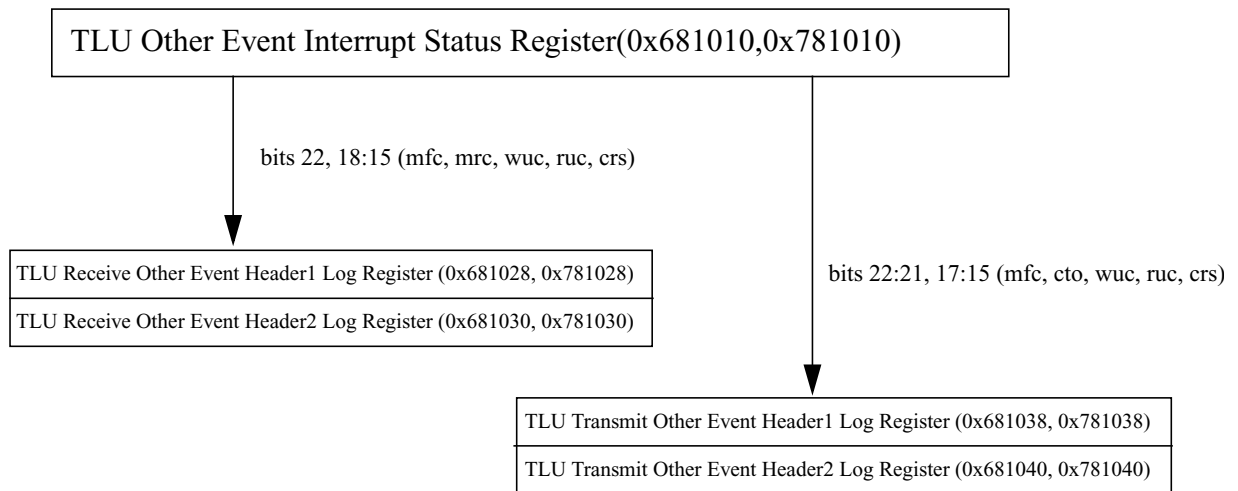
To clear any of the errors detected in the in TLU UE register, the TLU Correctable Error Status Clear Register (0x6a1018, 0x7a1018) should be used. To clear the duplicate registers in the LPU the LPU Link Layer Interrupt and Status Register (0x6e2210, 0x7e2210) and the LPU Phy Layer Interrupt and Status Register (0x6e2610, 0x7e2610) should be used for the appropriate duplicate bits.

PEC OE

If Mondo 62 was caused by a PEC OE error or event the next register that should be read is TLU Other Event Interrupt Status Register (0x681010,0x781010). This register shows all of the possible TLU OE errors and events which could have generated a

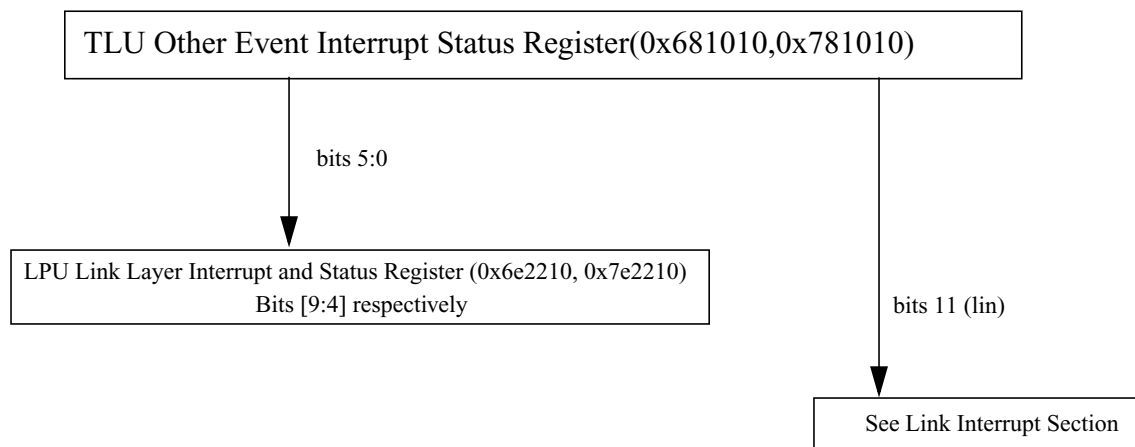
mondo 62. From these bits further information about the particular error can be obtained by further reading the proper Error logging register. Please see the diagram for details on these registers.

Figure 1-13 PEC OE Error Processing Diagram



Since this register, can store errors that are detected by the LPU as well as the TLU some of this information may be duplicated in the LPU in a different register. In the following diagram any bits which could store duplicate information are describe along with the register an the bit which would be duplicate.

Figure 1-14 PEC OE Duplicate Error Diagram

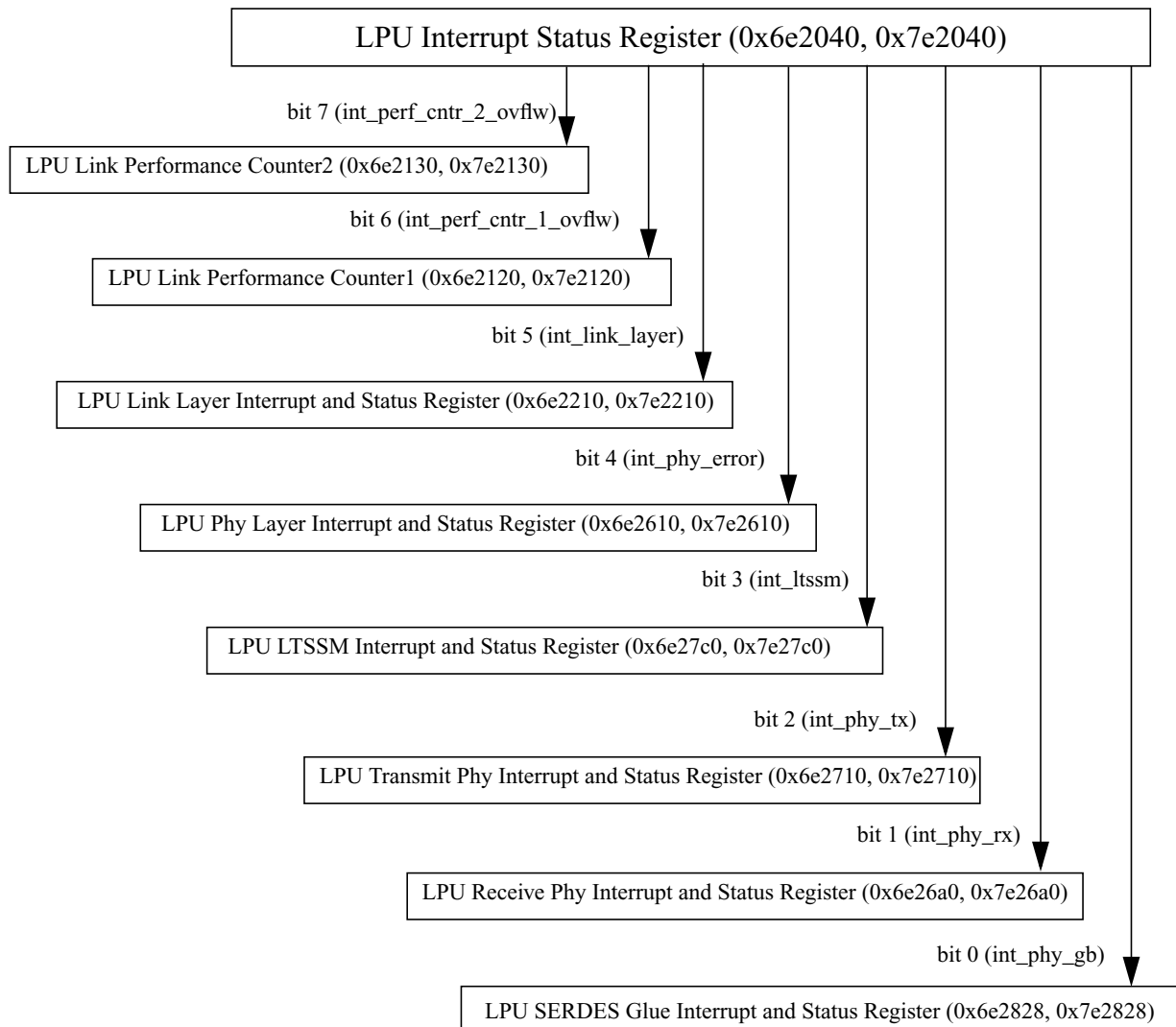


To clear any of the errors detected in the in TLU OE register, the TLU Other Event Status Clear Register (0x681018, 0x781018) should be used. To clear the duplicate registers in the LPU the LPU Link Layer Interrupt and Status Register (0x6e2210, 0x7e2210) should be used.

Link Interrupt

If Mondo 62 was caused by an Link Interrupt the next register that should be read is LPU Interrupt Status Register (0x6e2040, 0x7e2040). This register describes which of the one or more than one of the 8 possible block(s) in the LPU has an error or event which needs to be processed. From these bits further detail on the specific cause of the error can be found by further reading the proper interrupt and status register. Please see the diagram for details on these registers.

Figure 1-15 Link Interrupt Processing Diagram



To clear any of the errors detected by the link interrupt bit, the respective interrupt and status register shown above should be used.

Note – It should not be necessary to unmask link interrupts. They are present for diagnostic isolation and have corresponding notification in TLU errors.

1.5 Software Visible State

All flops in FIRE can be scanned out through a destructive JTAG operation. CSR's can also be accessed through a non-destructive JTAG operation. Internal RAM state cannot be accessed except where available via CSR accesses.

1.5.1 Accessing BIST Results

More information on accessing BIST results within Fire will be provided after layout.

1.6 Fire Error Injection

This section provides detail on how errors can be injected either internally within Fire, or on one of the external buses (JBus or PCIE).

1.6.1 JBus Error Injection

Fire allows the user to inject parity errors on JBus for Fire's outgoing address or data packets. Under software control via CSR writes to the "*JBus Parity Control Register*" errors will be injected on the JBus j_ad lines (the main address/data bus) by inverting the j_adp[3:0] bits associated with the j_ad lines. j_adp[3:0] covers the following bits:

- j_adp[0] -> j_ad[31:0]
- j_adp[1] -> j_ad[63:32]
- j_adp[2] -> j_ad[95:64]
- j_adp[3] -> {j_ad[127:96], j_adtype[7:0]}

One or more of the j_adp bits can be inverted under CSR control. The error can be injected on the "next" JBus Command/Address cycle or Data cycle, again selectable via CSR write. The next cycle is defined as the next valid JBus cycle driven by Fire after the CSR write to the parity control register matching the correct type specified by the user (either address or data cycle). The entire sequence of steps that the user should follow to inject an error is outlined below.

1. Perform a CSR write (using an NCWR) to the parity control register and set the following fields:
 - invert_par[5:2] - If invert_par[N] is set to 1, then j_adp[N] will be inverted when it is driven in the next valid cycle by Fire.
 - next_data - If set to 1, then the next valid cycle will be the next data cycle driven by Fire after the write to the parity control register is completed. This field is reset to 0 by hardware after the corrupted packet is driven to JBus.
 - next_addr - If set to 1, then the next valid cycle will be the next address cycle driven by Fire after the write to the parity control register is completed. This field is reset to 0 by hardware after the corrupted packet is driven to JBus.
2. Launch a transaction which will result in the correct JBus cycle type (address or data) being driven by Fire after the CSR write.

3. Repeat steps 1-3.

Note – If both `next_data` and `next_addr` are set by software at the same time then either the next address or data cycle, whichever occurs first, will have inverted parity associated with it. Both `next_data` and `next_addr` bits will then be cleared.

Fire does not provide any more means of insuring that a specific address/data cycle will be corrupted. Thus, it is the responsibility of the user to ensure the correct timing between the CSR write to the parity control register and the issuing of the packet to be corrupted by Fire. The timing of the write to the parity control register cannot be predicted if there is a lot of other additional activity within Fire so it is recommended that this error injection feature be used in a controlled testing environment. Otherwise, Fire's behavior will be very unpredictable.

PCI-E Error Injection

Fire allows users to inject or otherwise force errors on the PCI-E interface, the transport/link interface, and the MMU data cache.

1.6.1.1 MMU Data Cache Error Injection

The MMU Data Cache parity bits can be directly written, with good or bad parity, in the "MMU TTE Cache Data Registers" definition.

1.6.1.2 PCI-E Interface Error Injection/Force

1.6.1.2.1 Configure Flow Control Credits

The "TLU Ingress Credits Initial Register" will allow software to set how many credits will be advertised when the link is initialized for the following fields:

- Non-posted Header Credit
- Posted Header Credit
- Posted Data Credit

1.6.1.2.2 Ingress/Egress Control

The "TLU Diagnostic Register" enables the following diagnostic events, which can be examined in the "TLU Logged Other Event Status Register".

Force bad parity on the Ingress and Egress paths between the transport/link interfaces:

- Egress Parity Invert Bits
- Ingress Data Parity Invert Bits
- Ingress Header Parity Invert Bits

These are one-shot events enabled by 'triggers' in the same register:

- Egress Parity Invert Trigger
- Ingress Data Parity Invert Trigger
- Ingress Header Parity Invert Trigger

Memory Read Events can also be captured in the pipeline with a one-shot trigger which will cause the next memory read request to be captured in the header log registers and interrupt. The read request is not processed. This allows software to

process the request by reading the header log register, creating responses, writing them into the retry buffer, and setting the retry buffer transmission. Once a single memory read request is captured, hardware will reset this bit.

- Memory Read Capture Trigger

Egress packet processing and Ingress flow control updates can be disabled:

- Egress Packet Processing Disable
- Ingress Flow Control Update Disable

1.6.1.2.3 *Transmit Interface Diagnostic Control*

The "LPU Txlink Test Control Register" enables the following diagnostic events.

Force NAK event. When set will cause a Nak to be scheduled immediately (without waiting for Ack Nak latency timer). Hardware will reset this bit after a Nak is transmitted.

- Force Nak

Note – For the 'Force Nak' feature, it should only be used in a controlled environment as of that one or more TLPs are on the way to Fire from a remote device connected to Fire. Probably don't want to force Nak's on Idle cycles.

Force bad TLP CRC. When set, will cause the txlink hardware to invert CRC for the next TLP being transmitted. Hardware will reset this bit after the TLP transmission with inverted CRC is completed.

- Force bad TLP CRC

Force retransmission of TLP. When set will cause the txlink hardware to act as if a Nak was received with a sequence number same as the ackd_sequence number. This will cause retransmit of all TLPs in retry buffer. Hardware will reset this bit when retry is completed.

- Force Retransmit of TLP

Note – For the 'Force Retransmit of TLP' feature, it should only be used in a controlled environment as of: 1) retry buffer is NOT empty; 2) no Acks or Naks are being received; 3) no packets are being transmitted; 4) replay timer is not timing out.

Retry Buffer Software Access. Utilizing the Memory Address Control and Memory Data Load Registers in the PEC LPU, software has the capability to read and write data and parity in the retry buffer. This allows software to create a TLP. See the steps below for a more complete description.

Retry buffer write

- a. Set the RETRY_DISABLE bit in the LPU Link Layer Config Register.
- b. The 16 byte data to be loaded into a retry buffer location is first loaded into the memory data load register (Memory Data Load0,1,4)
LPU Txlink Memory Data Load0 Register: retry_buf{byte3, byte2, byte1, byte0}
LPU Txlink Memory Data Load1 Register: retry_buf{byte7, byte6, byte5, byte4}
LPU Txlink Memory Data Load4 Register: retry_buf{32'h0} (Note: bit[7:0] should be all 0s for automatic odd parity generation)

- c. Write the LPU Txlink Memory Address Control Register:
 - i. Address location is loaded into memory address location [15:0]
 - ii. FIFO select[28] is programmed 0 (retry FIFO select)
 - iii. Read/write select [29] is programmed 1 (write/select)
 - iv. Go [30] is asserted
- d. The 16 byte memory write takes place in retry buffer with parity calculated automatically. Once the memory write takes place, the Done bit[31] is asserted in the LPU Txlink Memory Address Control Register.
- e. Software clears Done [31] of the LPU Txlink Memory Address Control Register, by writing "1" to it.
- f. Steps a, b, c, d are repeated until a whole frame is programmed in retry buffer.

Sequence buffer write

- a. The sequence buffer should be loaded with {2b0, sequence number[11:0], retry pointer[15:0], eop position[1:0]} using a similar method to the retry buffer write, except that FIFO select[28] is programmed "1" (sequence FIFO select).
- b. The Retry buffer tail and write pointers and sequence buffer tail, and write pointers are loaded to reflect the new frame in the retry buffer.
- c. Software can now assert the Force Retransmit [0] bit in the "LPU Txlink Test Control Register" (mentioned above) to cause the buffer to be transmitted.

Retry buffer read

- a. Set the RETRY_DISABLE bit in the LPU Link Layer Config Register.
- b. Write the LPU Txlink Memory Address Control Register:
 - i. Address location is loaded into memory address location [15:0]
 - ii. FIFO select[28] is programmed 0 (retry FIFO select)
 - iii. Read/write select [29] is programmed 0 (read select)
 - iv. Go [30] is asserted
- c. The Read data appears in the memory data register and the Done bit [31] gets asserted.
LPU Txlink Memory Data Load0 Register: retry_buf{byte3, byte2, byte1, byte0}
LPU Txlink Memory Data Load1 Register: retry_buf{byte7, byte6, byte5, byte4}
LPU Txlink Memory Data Load4 Register[7:0]: retry_buf{rbuf_parity[0:7]}
- d. Software clears Done [31] of Memory Control register, by writing "1" to it.
- e. Software repeats steps a,b,c until the whole frame is read out of retry buffer.

Sequence buffer read

- a. Sequence buffer read can be done in a similar manner, except that FIFO select[28] is programmed "1" (sequence FIFO select).

Programming and Sending a TLP from the Retry Buffer

- a. Program the retry buffer following the Retry Buffer write instructions.
- b. Program the sequence buffer following the Sequence Buffer write instructions.
- c. Set the MSK_RTRY_UNF_OVF and MSK_RTRY_BUF_OVF bit the LPU Link Layer Interrupt Mask Register.
- d. Set the RTRY_FIFO_TLPTR and RTRY_FIFO_HDPTR in the LPU Txlink Retry FIFO Pointer Register.
- e. Set the SEQ_CNT_TLPTR and SEQ_CNT_HDPTR in the LPU Txlink Sequence Count FIFO Pointers Register.
- f. Set the RTRY_DATA_CNT in the LPU Txlink Retry Data Count Register.
- g. Set the SEQ_BUFF_CNT in the LPU Txlink Sequence Buffer Count Register.
- h. Write the LPU Txlink Sequence Buffer Bottom Data Register.
- i. Set the WE, ACK_SEQ_CNTR, NXT_TX_SEQ_CNTR in the LPU Txlink Sequence Counter Register.
- j. Clear the RTRY_DISABLE in the LPU Link Layer Config Register.
- k. Clear all the interrupts in the LPU Link Layer Interrupt and Status Register.
 - l. Unmask the interrupts in the LPU Link Layer Interrupt Mask Register.
- m. Set the FORCE_RTX_TLP in the LPU Txlink Test Control Register.

Note – For the ‘Retry buffer write’ feature, it should only be used in a controlled environment as of that the traffic is quiesced prior applying the feature and the link is reset to straight out flow control credits after the retry buffer write is done.

OBP example using Fire’s Leaf B registers in a non-Niagara system

The following OBP code example illustrates how to set up the Retry Buffer and Sequence Buffer to send a TLP from Fire’s PCI-Express Leaf B.

```
: force-retry-test ( -- )
\ a)
0 01000020 0f000000 0 retry-buf-wr-b
0 retry-buf-rd-b

8 00000000 00001000 0 retry-buf-wr-b
8 retry-buf-rd-b

10 01000020 0f000000 0 retry-buf-wr-b
```

10 retry-buf-rd-b

18 00000000 10001000 0 retry-buf-wr-b

18 retry-buf-rd-b

\ b)

0 0 f 1 seq-buf-wr-b

0 seq-buf-rd-b

1 1 1f 1 seq-buf-wr-b

1 seq-buf-rd-b

\ c)

\ Write LPU Link Layer Interrupt Mask Register, MSK_RTRY_UNF_OVF 9,

\ MSK_RTRY_BUF_OVF 8

400.0f7E2220 15 spacex@ 300 or 400.0f7E2220 15 spacex!

\ Read LPU Link Layer Interrupt Mask Register

400.0f7E2220 15 spacex?

\ d)

\ Write LPU Txlink Retry FIFO Pointer Register, RTRY_FIFO_TLPTR 31:16

\ RTRY_FIFO_HDPTR 15:0

1f.0000 400.0f7E2430 15 spacex!

\ e)

\ Write LPU Txlink Sequence Count FIFO Pointers Register, SEQ_CNT_TLPTR 27:16

\ SEQ_CNT_HDPTR 11:0

1.0000 400.0f7E2460 15 spacex!

\ f)

\ Write LPU Txlink Retry Data Count Register, RTRY_DATA_CNT 15:0

20 400.0f7E24C0 15 spacex!



\ g)

\ Write LPU Txlink Sequence Buffer Count Register, SEQ_BUFF_CNT 11:0

2 400.0f7E24C8 15 spacex!

\ h)

\ Write LPU Txlink Sequence Buffer Bottom Data Register,

3d 400.0f7E24D0 15 spacex!

\ i)

\ Write LPU Txlink Sequence Counter Register, WE 31, ACK_SEQ_CNTR 27:16,

\ NXT_TX_SEQ_CNTR 11:0

8000.0002 400.0f7E2448 15 spacex!

\ j)

\ Write LPU Link Layer Config Register, RETRY_DISABLE 1

400.0f7E2200 15 spacex@ ffff.ffff and 400.0f7E2200 15 spacex!

\ Read LPU Link Layer Config Register, RETRY_DISABLE 1

400.0f7E2200 15 spacex?

\ k)

\ Write LPU Link Layer Interrupt and Status Register

ffff.ffff 400.0f7E2210 15 spacex!

\ Write LPU Link Layer Interrupt Mask Register

0 400.0f7E2220 15 spacex!

\ l)

cr

." Press any key to start FORCE_RTX_TLP" cr

begin key? until

\ Write LPU Txlink Test Control Register, FORCE_RTX_TLP 0


```
400.0f7E2470 15 spacex@ 1 or 400.0f7E2470 15 spacex!
;

: retry-buf-rd-b ( addr -- )
." Firmware read of retry buffer, addr: " 0 pick. cr

\ Write LPU Link Layer Config Register, RETRY_DISABLE(1)
400.0f7E2200 15 spacex@ 2 or 400.0f7E2200 15 spacex!

\ Write LPU Txlink Memory Address Control Register
\ go bit
h# 1 d# 30 lshift
\ rd/wr
h# 0 d# 29 lshift
\ fifo_select
h# 0 d# 28 lshift
\ retry buf addr first item on stack
or
400.0f7E2480 15 spacex!

\ Read LPU Txlink Memory Address Control Register
." Memory Address Control Reg value: " 400.0f7E2480 15 spacex@ . cr

\ Wait for done bit
begin 400.0f7E2480 15 spacex@ 8000.0000 and until

\ Clear done bit
8000.0000 400.0f7E2480 15 spacex!

\ Read regs
." Reg0: " 400.0f7E2488 15 spacex@ . cr
." Reg1: " 400.0f7E2490 15 spacex@ . cr
```

```
. " Reg4: " 400.0f7E24a8 15 spacex@ . cr
;

: retry-buf-wr-b ( addr data0 data1 data4 -- )
. " Firmware write of retry buffer, addr: " 3 pick. cr

\ Write LPU Link Layer Config Register, RETRY_DISABLE(1)
400.0f7E2200 15 spacex@ 2 or 400.0f7E2200 15 spacex!

\ Write regs
2 pick 400.0f7E2488 15 spacex!
1 pick 400.0f7E2490 15 spacex!
0 pick 400.0f7E24a8 15 spacex!

\ Read regs
. " Reg0: " 400.0f7E2488 15 spacex@ . cr
. " Reg1: " 400.0f7E2490 15 spacex@ . cr
. " Reg4: " 400.0f7E24a8 15 spacex@ . cr

\ Write LPU Txlink Memory Address Control Register
\ retry buf addr
3 pick
\ go bit
h# 1 d# 30 lshift
\ rd/wr
h# 1 d# 29 lshift
\ fifo_select
h# 0 d# 28 lshift
or
400.0f7E2480 15 spacex!

\ Read LPU Txlink Memory Address Control Register
```

```
." Memory Address Control Reg value: " 400.0f7E2480 15 spacex@ . cr
```

```
\ Wait for done bit
```

```
begin 400.0f7E2480 15 spacex@ 8000.0000 and until
```

```
\ Clear done bit
```

```
8000.0000 400.0f7E2480 15 spacex!
```

```
\ Clear stack of our items
```

```
4 0 do drop loop
```

```
;
```

```
: seq-buf-rd-b ( addr -- )
```

```
." Firmware read of seq buffer, addr: " 0 pick. cr
```

```
\ Write LPU Link Layer Config Register, RETRY_DISABLE(1)
```

```
400.0f7E2200 15 spacex@ 2 or 400.0f7E2200 15 spacex!
```

```
\ Write LPU Txlink Memory Address Control Register
```

```
\ go bit
```

```
h# 1 d# 30 lshift
```

```
\ rd/wr
```

```
h# 0 d# 29 lshift
```

```
\ fifo_select
```

```
h# 1 d# 28 lshift
```

```
\ retry buf addr first item on stack
```

```
or
```

```
400.0f7E2480 15 spacex!
```

```
\ Read LPU Txlink Memory Address Control Register
```

```
." Memory Address Control Reg value: " 400.0f7E2480 15 spacex@ . cr
```

```
\ Wait for done bit
begin 400.0f7E2480 15 spacex@ 8000.0000 and until

\ Clear done bit
8000.0000 400.0f7E2480 15 spacex!

\ Read reg
400.0f7E2488 15 spacex@
." odd parity: " dup d# 30 rshift h# 1 and . cr
." seq num:  " dup d# 18 rshift h# fff and . cr
." retry ptr: " dup d# 2 rshift h# ffff and . cr
." eop pos:  " dup h# 3 and . cr
drop
;

: seq-buf-wr-b ( addr seq_num retry_ptr eop_pos --)
." Firmware write of seq buffer, addr: " 3 pick. cr

\ Write LPU Link Layer Config Register, RETRY_DISABLE(1)
400.0f7E2200 15 spacex@ 2 or 400.0f7E2200 15 spacex!

\ Write reg
\ seq_num
2 pick d# 18 lshift
\ retry_ptr
2 pick d# 2 lshift
\ eop_pos
2 pick
or
dup ." Data Load0 Reg value: " . cr

400.0f7E2488 15 spacex!
```

```
\ Read regs
.\" Reg0: " 400.0f7E2488 15 spacex@ . cr

\ Write LPU Txlink Memory Address Control Register
\ retry buf addr
3 pick
\ go bit
h# 1 d# 30 lshift
\ rd/wr
h# 1 d# 29 lshift
\ fifo_select
h# 1 d# 28 lshift
or
400.0f7E2480 15 spacex!

\ Read LPU Txlink Memory Address Control Register
.\" Memory Address Control Reg value: " 400.0f7E2480 15 spacex@ . cr

\ Wait for done bit
begin 400.0f7E2480 15 spacex@ 8000.0000 and until

\ Clear done bit
8000.0000 400.0f7E2480 15 spacex!

\ Clear stack of our items
4 0 do drop loop
;
```